Improvement in Error Performance of Turbo Codes

by
BHUWANENDRA KUMAR

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
March, 2000

Improvement in Error Performance of Turbo Codes

A Thesis Submitted

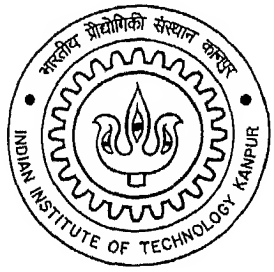
In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF TECHNOLOGY

By

BHUWANENDRA KUMAR

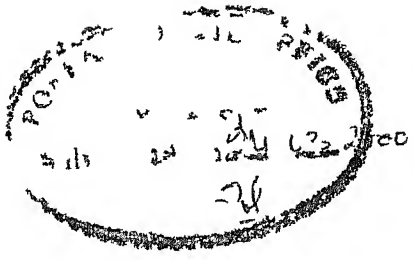


to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

MARCH 2000



Certificate

This is to certify that the work contained in the thesis entitled ***Improvement in Error Performance of Turbo Codes*** has been carried out by Bhuwanendra Kumar (Roll No 9810413) under my supervision and that this work has not been submitted elsewhere for any degree

Vishwanath Sinha
Professor
Dept of Electrical Engineering
Indian Institute of Technology
Kanpur – 208016

March 2000

Abstract

In recent years iterative concatenated decoding has regained popularity starting with the remarkable results presented in a paper by a group of French researchers They introduced a new family of convolutional codes nicknamed Turbo codes after the resemblance with the turbo engine A turbo code is built from parallel concatenation of two recursive systematic codes linked together by nonuniform interleaving Decoding is done iteratively by two maximum a posteriori probability decoders each using the decoding results from other one For sufficiently large interleaver size the error performance seems to be close to Shannon limit

In this thesis we examine the performance of turbo codes on the additive white Gaussian noise channel The influence of the size of encoder memory different type and size of interleaver on decoding is examined A new dynamic decoding algorithm is proposed which uses variable number of iterations A modification to odd even interleaver has been suggested Principle of turbo code has been extended to an encoder which uses three recursive systematic convolutional encoders We show that odd even interleaver performs better after modification We also show that dynamic algorithm requires less time as compared to the other algorithm for achieving the same error performance and that the extended turbo code can achieve better results with small interleaver size

Dedicated to My Parents and My Sister Gayatri

Acknowledgement

I take this opportunity to express my sincere gratitude to my guide Dr V Sinha for the benevolent guidance and motivation provided by him I would like to thank him for providing me a free atmosphere of work which doubled my motivation His faith and confidence in me acted as a catalyst for my work efficiency His knowledge about my limitations abilities and interests made me work with my full efficiency I would also like to thank to all my teachers who introduced me the fundamentals of Communications

I would also like to cherish the nice company of my friends Dhaval Alpana Nitin Praveen Harish Rao Murthy and Gaurav without which the life in the lab would have been monotonous Thanks to my friends Asesh Raja and Dipan for their help and encouragement I am also grateful to Vineet Vivek and Manoj for providing me various kinds of help in the lab during the course of my thesis work

I must not forget to mention the name Kalpesh and Sibu my friends who were always beside me throughout the M Tech programme Last but not least I wish to acknowledge the constant encouragement and blessings which my parents gave to me

Contents

List of Figures

List of Tables

List of Abbreviations

APP	A *posteriori* probability
AWGN	Additive white gaussian noise
BER	Bit error rate
BPSK	Binary phase shift keying
MAP	Maximum a *posteriori*
ML	Maximum likelihood

Chapter 1

Introduction

In a digital communication systems information is represented as a sequence of binary bits After certain processing the resultant binary bits are modulated to analog signal waveform before transmission over a communication channel The communication channel introduces noise and interference corrupting the transmitted waveform At the receiver the corrupted received waveform is mapped back to the binary bits after due estimation process Errors occur in the bit stream due to the transmission impairments and the number of errors depends on the amount of noise and interference in the communication channel

Channel coding is often used in digital communication system to protect the digital information from noise and interference by correcting a number of bit errors that occur due to the channel impairment Channel coding is accomplished mostly by selectively introducing redundant bits into the transmitted information stream These additional bits facilitate detection and correction of bit errors in the received data stream and provide more reliable information transmission Naturally channel coding has become an area of increasing importance in communications The reasons for this are mainly its usefulness in preserving the data integrity and its potential of providing tradeoffs in a communication system design In any system that handles large amount of data uncorrected and undetected errors can degrade performance response time and possibly increase the need for human intervention The coding can change data quality from problematic to acceptable If a communication system has no problem with data quality the designer can think of discarding or downgrading the most expensive and troublesome elements and overcome the resulting loss in performance by using an appropriate error correcting coding In this sense the error control is a system design technique that can fundamentally change the tradeoff in a communication system design

1 1 Types of Channel Codes

There are two main types of channel codes namely block codes and convolution codes There is much difference between block codes and convolution codes Block codes are based rigorously on finite field arithmetic and abstract algebra They can be used either to detect or correct error or both Block codes accept a block of k information bits and produce a block of n coded bits By predetermined rules n k bits are added to the k information bits to form the n coded bits Commonly these codes are referred as (n k) block code Some of the commonly used block codes are binary Hamming codes Golay codes BCH codes and non binary Reed Solomon codes Several ways exit to decode block codes and estimate the information bits These decoding techniques will not be discussed here [20]

Convolution codes are one of the most widely used channel codes in the practical communication system These codes are developed with a separate mathematical structure and are primarily used in real time error correction Convolution codes convert the entire data stream into one single code word The encoded bits depend not only current k input bits but also on past input bits The main decoding strategy is based on the widely used Viterbi algorithm [20]

1 2 Racing Towards the Shannon Limit

The history of error control coding and information theory began in 1948 when Claude E Shannon published his famous paper A Mathematical theory of Communication In his paper Shannon showed that every communication channel has a parameter C (measured in bits per second) called the channel capacity and one can transmit digital information through the channel at a rate up to the channel capacity with arbitrarily small probability of error The minimum value of bit energy to noise ratio (E_b/N_0) to achieve this capacity limit is −1 59 dB and known as a Shannon limit

We know from the coding theory that increasing the code word length or encoder memory greater protection or coding gain can be achieved At the same time complexity of maximum likelihood (ML) decoding algorithm increases exponentially and the algorithm becomes difficult to implement The increased error correction capability of long block codes or convolution codes requires a very high computational

effort This led to research for new coding schemes which could replace ML decoding algorithm with simpler decoding strategies Using these iterative technique one can approach the performance of ML decoding algorithm The encoding technique used in iterative decoding are those coding techniques which combine different code in such a way that each of them can be decoded independently The codes in this category are product block codes and concatenated codes A common feature of these techniques is that decoding can be done sequentially using one decoder at a time (i e one code is decoded another decoder is used for the next code and so on)

Product codes were first attempted for achieving higher coding gain without requiring the decoding complexity of long codes Unfortunately the iterative decoding of the component codes did not produce the improvement in performance as was expected even though the error correction power of the product code increased as a whole [20]

Better results were obtained with concatenated codes The component codes are called the inner and outer codes First the inner code vector is decoded then decoded bits are again decoded according the outer code used All these codes have disadvantage that no information is exchanged between the two encoders

In 1993 Berrou et al introduced a new coding scheme which has been nicknamed as Turbo Code [1] It was one of the most important developments in coding theory for many years It has been shown that a turbo code can achieve performance within 1 dB of Shannon limit The main advantage of these codes when used together with iterative decoding scheme is low complexity in spite of high performance which makes it suitable for any application which requires lower signal to noise ratio It is therefore part of the standard for the third generation mobile telecommunication system

1 3 A Brief Overview of the Turbo Code

The turbo code encoder is a parallel concatenation of two recursive systematic convolutional (RSC) encoders called the constituent encoders The information block which has the finite length is encoded by first constituent encoder but interleaved before it enters the second constituent encoder Note that interleaver operates on an information sequence and not on a code sequence The interleaver size equals the

information block length The output of the constituent encoders is generally punctured to increase the code rate

The encoder is used in conjunction with a suboptimal decoding scheme based on a modified version of the maximum a posteriori (MAP) algorithm introduced by Bahl et al [3]

1 4 Objective of the Thesis

The objective of the thesis is to study the performance of turbo codes in the additive white gaussian noise (AWGN) environment Various parameter affect the performance of the turbo codes these are memory of the constituent encoders number of iterations used in the decoding process puncturing and choice of interleaver Effects of all these parameters have been studied through simulation During the decoding of the turbo coded information blocks some blocks require less number of iterations to become error free whereas some error prone blocks require more number of iterations decoding algorithm with fixed number of iteration keeps on iterating a block till the last iteration even it become error free after few iterations so a dynamic algorithm with variable number of iteration has been proposed Relative performance of various interleavers has been studied The principle of turbo code has been extended to an encoder which uses three encoder and two interleavers

1 5 Organization of the Thesis

In this thesis a new class of convolution codes which is called turbo code is presented Chapter 2 introduces the basic turbo code encoder and decoder The turbo code encoder is parallel concatenation of two recursive systematic convolution (RSC) codes separated by an interleaver This chapter shows the construction of RSC encoder from a nonrecursive nonsystematic (conventional) convolution encoder It discusses the similarities and difference between the conventional and RSC encoder in terms of their intrinsic properties Furthermore this chapter also describes the structure of the turbo code decoder Chapter 3 discusses the function of interleaver and describes many different types of interleavers that are suitable for turbo code encoder Chapter4 investigates the performance of turbo codes through extensive computer simulation

The simulation setup is carefully detailed Many simulation results are then presented to show the important characteristics of turbo code Chapter 5 then summarizes the important finding about turbo codes and concludes the thesis

Chapter 2

Turbo Code Encoder and Decoder

This chapter describes the turbo code encoder and decoder and their components in detail The well known turbo code encoder is built using two identical recursive systematic convolutional (RSC) codes with parallel concatenation [1] An RSC encoder is typically a $r=\frac{1}{2}$ code and is termed a component encoder An interleaver separates the two component encoder Only one of the systematic outputs from the two component encoders is used because the systematic output from the other component encoder is just a permuted version of the chosen systematic output Fig 2 1 shows the fundamental turbo encoder

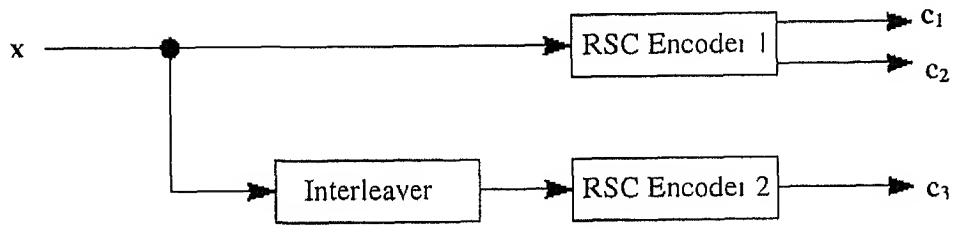


Fig 2 1 Fundamental turbo code encoder

2 1 Convolutional Encoder

A convolution code introduce the redundant bits into the data stream through the use of linear shift Registers as shown in the fig 2 2

The information bits are input into the shift registers and output encoded bits are obtained by modulo 2 addition of the input information bits and the contents of shift registers at various stages of delay The connections of modulo 2 adders were initially developed heuristically with no algebraic or combinatorial foundation

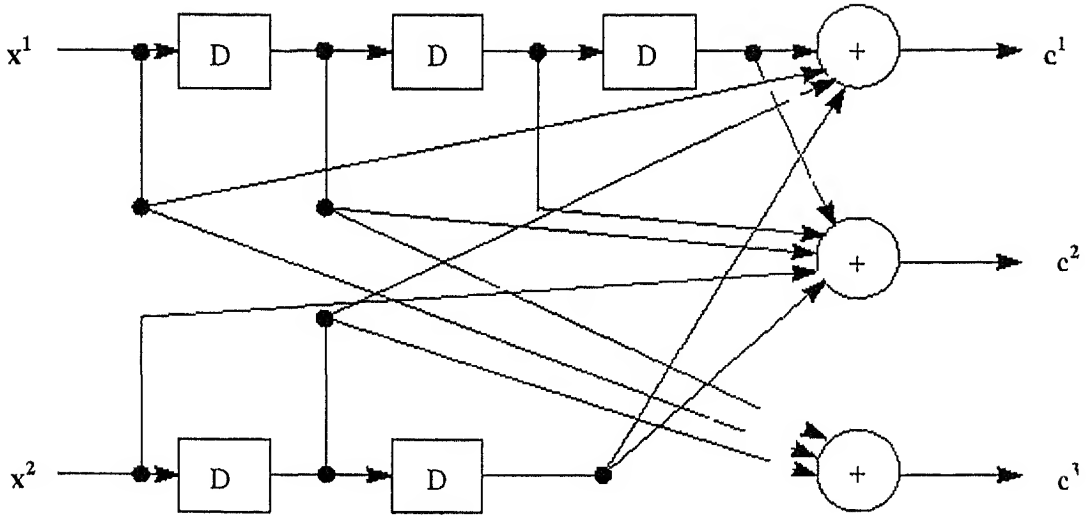


Fig 2 2 Example of convolution encoder with x^i as input information bit stream and c^i as an output encoded bit stream

The code rate for a convolution code is defined as

$$r = \frac{k}{n} \tag{2 1}$$

where k is the number of parallel input information bits and n is the number of parallel output encoded bits at one time interval The constraint length K for a convolution code is defined as

$$K = v + 1 \tag{2 2}$$

where v is the maximum number of stages (memory size) in any shift register The shift register stores the state information of the Convoltuional encoder and the constraint length relates the number of bits upon which the output depends For the convolutional encoder shown in fig 2 2 the code rate r=2/3 the maximum memory size v =3 and the constraint length K=4

2 2 Recursive Systematic Convolutional (RSC) Encoder

Figure 2 3 shows a conventional convolutional encoder This convolutional encoder is represented by the generator sequences g_1=[1 1 1] and g_2=[1 0 1] and can be

equivalently represented in a more compact form as G=[g_1 g]

The recursive systematic convolutional (RSC) encoder is obtained from the nonrecursive nonsystematic (conventional) convolutional encoder by feeding back one of its encoded outputs to its input

The RSC encoder of this conventional convolutional encoder is represented as G=[1 g_2/g_1] where the first output (represented by g_1) is fed back to the input In the above representation 1 denotes the systematic output and g_1 is fed back to the input of the RSC encoder Fig 2 4 shows the resulting RSC encoder

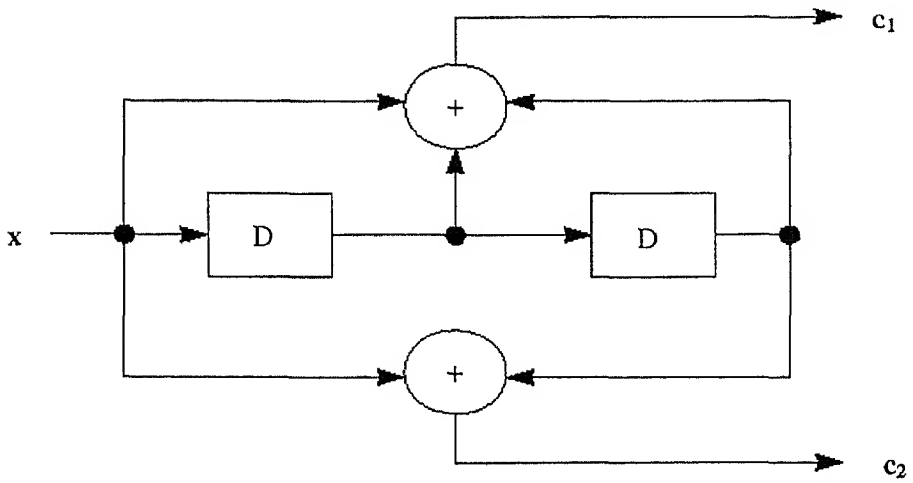


Fig 2 3 Conventional convolutional encoder with r=1/2 and K=3

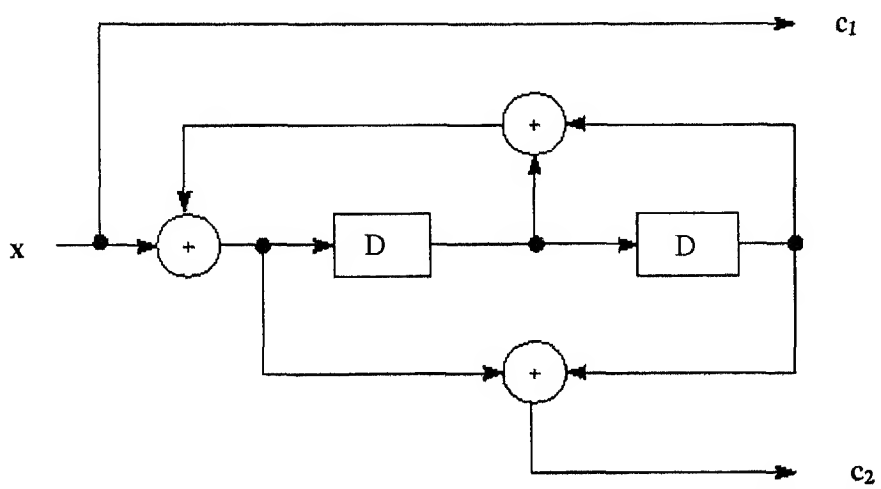


Fig 2 4 The RSC encoder obtained from fig 2 3 with r=1/2 and K=3

2 2 1 Trellis Termination

For the conventional convolutional encoder trellis is terminated by inserting v=K 1 additional zero bits after the input block These additional bits drive the conventional convoltutional encoder to the all zero state (trellis termination) However this strategy is not possible in RSC encoder due to the feedback The additional termination bits for the RSC encoder would depend on the state of the encoder at the end of input block and are naturally very difficult to predict Furthermore if the termination bits for the first encoder are found other component encoder cannot be driven into the all zero state due to the piesence of interleaver Fig 2 5 shows a scheme that has been developed in [17] for terminating the RSC encoder

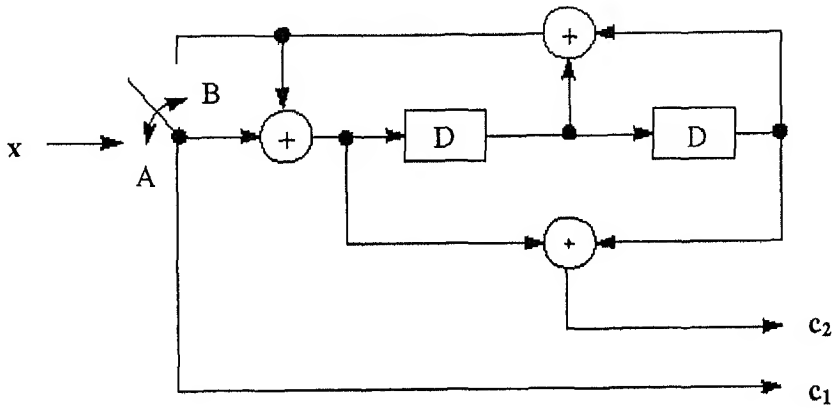


Fig 2 5 Trellis termination scheme for RSC encoder

For encoding the input sequence the switch is turned on to position A and for terminating the trellis the switch is turned on to position B at the end of input block

2 3 Recursive and Nonrecursive Convolution Encoder

The nature of recursive and non recursive convolution encoder is best examined by an example Fig 2 6 shows a simple non recursive convolution encoder with generator sequences g_1=[1 1] and g_2=[1 0] Fig 2 7 shows the equivalent recursive encoder of fig 2 6 with G =[1 g_2/g_1]

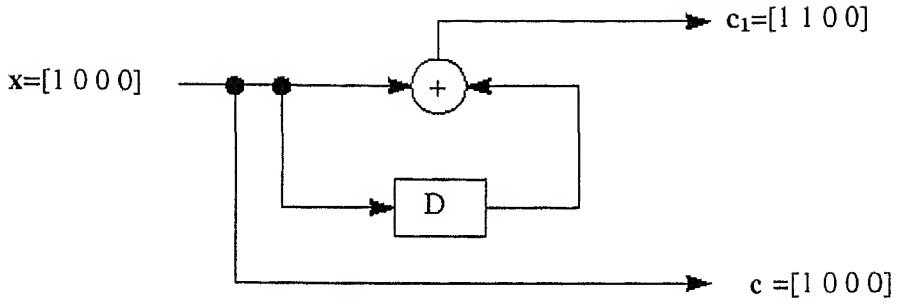


Fig 2 6 Non recursive r=1/2 and K=2 convolutional encoder with input and output sequences

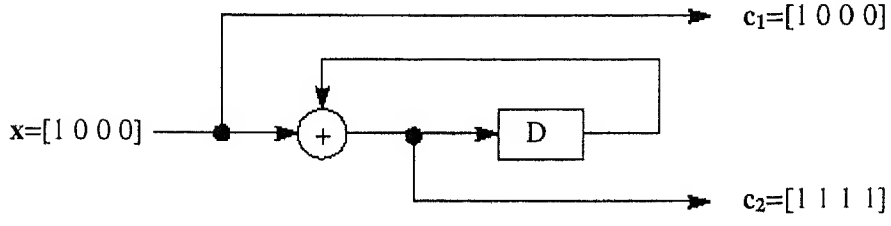


Fig 2 7 Recursive r=1/2 and K=2 convolutional encoder of fig 2 6 with input and output sequences

In the above figures flow of input bits is from left to right From fig 2 6 and fig 2 7 given the same input sequence the non recursive encoder produces an output code word with weight of 3 and the recursive encoder produces an output code word with weight of 5 In general a recursive convoltutional encoder tends to produce codewords with increased weight relative to a non recursive encoder This results in fewer code words with lower weights and this leads to better error performance For Turbo codes the main purpose of the RSC encoders as component encoders is to utilize the recursive nature of the encoders and not the fact that encoders are systematic Fig 2 8 shows the state diagram of the non recursive encoder

Fig 2 9 shows the state diagram of recursive encoder Clearly the state diagrams of the encoders are very similar Furthermore two codes have the same minimum free distance and can be described by the same trellis structure [8]

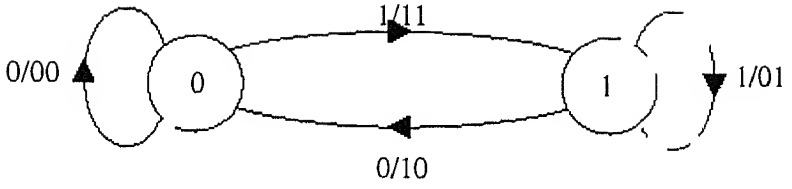


Fig 2 8 State diagram of nonrecursive encoder in fig 2 6

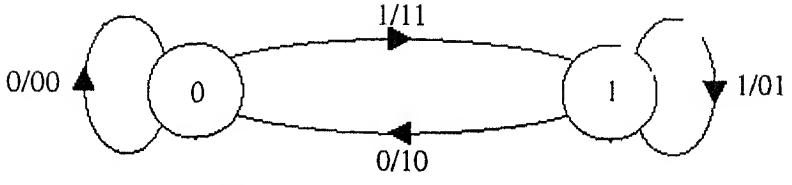


Fig 2 9 State diagram of recursive encoder in fig 2 7

2 4 Concatenation of Codes

A concatenated code is composed of two separate codes that are combined to form a large code [19] There are two types of concatenation namely serial and parallel concatenations Fig 2 10 shows the serial concatenation scheme for transmission

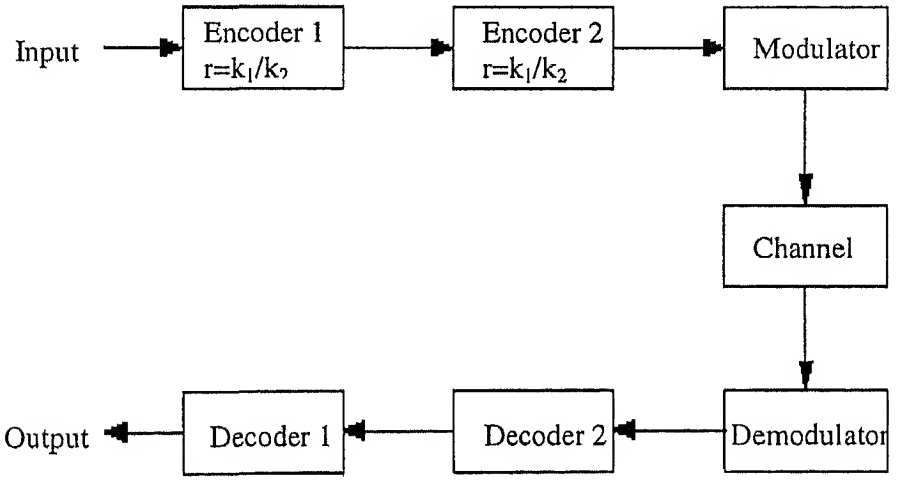


Fig 2 10 Serial concatenated code

The total code rate for serial concatenation is

$$r_{t,s} = \frac{k_1 k}{n_1 n} \tag{2 3}$$

which is equal to the product of two codes rate Fig 2 11 shows a parallel concatenation scheme for transmission

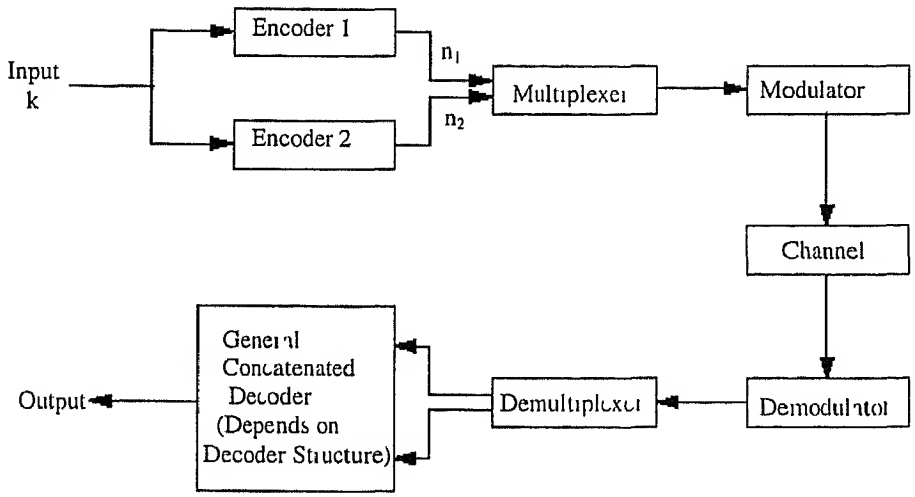


Fig 2 11 Parallel concatenated code

The total code rate for parallel concatenated code is

$$r_t = \frac{k}{n_1 + n} \tag{2 4}$$

For both serial and parallel concatenated schemes an interleaver is often used between the encoders to improve the burst error correction capability or to increase the randomness of the code

Turbo codes use parallel concatenated coding scheme However the turbo code decoder is based on the serial concatenated decoding scheme The serial concatenated decoders are used because they perform better than the parallel concatenated decoding due to the fact that the serial concatenation scheme has the ability to share the information between the decoders whereas the decoders for the parallel concatenated coding scheme primarily decode independently

2 5 Puncturer

Puncturing is often used in order to achieve high coding rates When puncturing is used some of the parity bits generated by component encoder1 and component

encoder2 are periodically deleted according to a chosen perforation pattern defined by a matrix P For a global rate of R=1/2 P is chosen as

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

This means that parity bits from encoder1 are sent every odd time and parity bits from encoder2 are sent every even time Fig 2 12 shows a standard turbo encoder

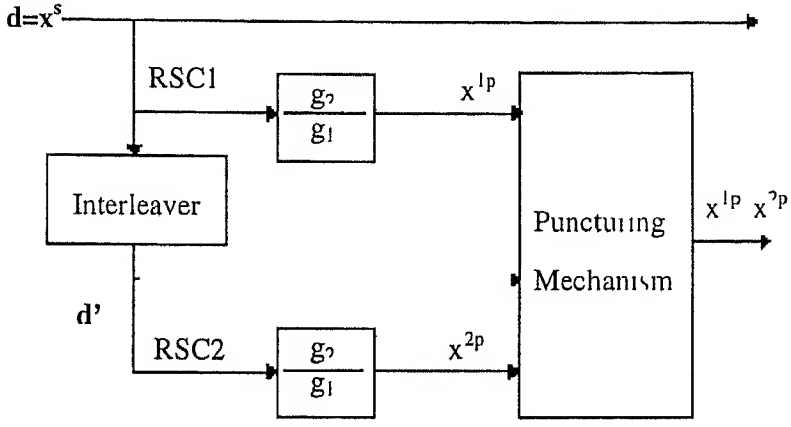


Fig 2 12 Standard turbo code encoder

2 6 The Decoder

Just prior to discovery of turbo codes there was much interest in the coding community in suboptimal decoding strategies for concatenated codes involving multiple (usually two) decoders operating cooperatively Most of the focus was on a type of Viterbi decoder which provides soft output (or reliability) information to a companion soft output Viterbi decoder for use in a subsequent decoding Recently some attention was the symbol by symbol maximum a posterior (MAP) algorithm of Bahl (BCJR algorithm) et al [ ] published over 20 years ago It was the later algorithm often called the BCJR algorithm that Berrou et al utilized in the iterative decoding of turbo codes We discuss the BCJR algorithm employed by each of the constituent encoders We first discuss a modified version of the BCJR algorithm for performing

symbol by symbol MAP decoding we then show how this algorithm is incorporated into an iterative decoder employing two BCJR MAP decoders

2 6 1 The Modified BCJR Algorithm

In the following derivation we consider rate half RSC Let **S** be the set of all states of RSC encoder The input to the encoder is data sequence **d** =(d_1 d_N) The outputs of the encoder are systematic data bit sequence **x** =(x_1 x_N)=**d** and parity bit sequence $\mathbf{x}^p$=(x_1^l x_N^l) These output bits are modulated with a BPSK modulator and sent over an AWGN channel At the receiver the log likelihood ratio L (d_k) is defined as

$$L(d_k) = \log \frac{P(d_k = 1 \mid observation)}{P(d_k = 0 \mid observation)} \qquad (2\,5)$$

Where P_r (d_k= r|observation) r=0 1 is the a posteriori probability (APP) of data bit d_k The encoder output at time k is translated into +1/ 1 values by the BPSK modulator

$$a_k = 2d_k \; 1$$

$$b_k = 2x_k^l \; 1$$

The pair (a_k b_k) defines the transmitted symbol C_k at time k

The sequence of transmitted symbol is given by

$$\mathbf{C}_1^N = (C_1 \; C_2 \quad C_N) \qquad (2\,6)$$

here the subscripts denotes the time index of bit in a particular block of size N Which is the input to a discrete Gaussian memoryless channel whose output sequence is defined as

$$\mathbf{R}_1^N = (R_1 \; R_2 \quad R_N) \qquad (2\,7)$$

with R_k=(y_k, y_k^l) being the received symbol at time k y_k and y_k^l are defined as

$$y_k = (2d_k \; 1) + p_k \qquad (2\,8)$$

$$y_k^l = (2x_k^l \; 1) + q_k \qquad (2\,9)$$

where p_k and q_k are two independent normal variables with variance σ^2 The APP of decoded data bit d_k can be derived from the joint probability defined by

$$\lambda_k(m) = P(d_k = i, S_k = m \mid \mathbf{R}_1^N) \tag{2.10}$$

With i=0 1 and m=0 1 2^v 1 The APP of decoded data bit d_k is thus equal to

$$P(d_k = i \mid \mathbf{R}_1^N) = \sum \lambda_k(m) \tag{2.11}$$

where the summation is over all m=0 1 2^v 1 From (2 5) and (2 11) the L (d_k) associated with a decoded bit d_k can be written as

$$L(d_k) = \log \frac{\sum \lambda_k^1(m)}{\sum \lambda_k^0(m)} \tag{2.12}$$

This represents the soft output of MAP decoder This can be used as input to another decoder in a concatenated scheme or in the next iteration in an iterative decoder Finally the decoder can make a hard decision by comparing L (d_k) to a threshold equal to zero

if $L(d_k) \geq 0$ the decoded bit is 1

if $L(d_k) < 0$ the decoded bit is 0

In order to compute the APP of decoded data bit d_k as in (2 11) it is useful to define the following probability functions

$$\alpha_k(m) = P_i(S_k = m \mid \mathbf{R}_1^k) \tag{2.13}$$

$$\beta_k(m) = P_i(\mathbf{R}_{k+1}^N \mid S_k = m) \frac{1}{P(\mathbf{R}_{k+1}^N \mid \mathbf{R}_1^k)} \tag{2.14}$$

$$\gamma_i(R_k \; m \; m) = P(d_k = i \; R_k \; S_k = m \mid S_{k-1} = m) \tag{2.15}$$

Where state m at time k is denoted by S_k=m

Recalling Bayes rule

$$P(B \mid A) = \frac{P(A \mid B) \; P(B)}{P(A)}$$

and the expression for the conditional probability

$$P(B \mid A) = \frac{P(A \; B)}{P(A)}$$

Combining these two equalities we obtain the following well known relation

$$P(A \; B) = P(A \mid B) \; P(B) \tag{2.16}$$

which turns out to be very useful for the following derivation

After applying the (2 16) in the expression (2 11) it becomes

$$P(d_k = i \mid \mathbf{R}_1^N) = \sum_m P(d_k = i, S_k = m \mid \mathbf{R}_1^N) = \sum_m P(d_k = i\ S_k = m, \mathbf{R}_1^N)\ \frac{1}{P(\mathbf{R}_1^N)}$$

Since the observations are independent of time this can be rewritten as

$$\sum_m P(d_k = i, S_k = m, \mathbf{R}_1^N)\ \frac{1}{P(\mathbf{R}_1^k, \mathbf{R}_{k+1}^N)}$$

Applying (2 16) to the numerator and denominator gives

$$\frac{\sum_m P(\mathbf{R}_{k+1}^N \mid d_k = i, S_k = m, \mathbf{R}_1^k)\ P(d_k = i, S_k = m\ \mathbf{R}_1^k)}{P(\mathbf{R}_{k+1}^N \mid \mathbf{R}_1^k)\ P(\mathbf{R}_1^k)}$$

Using the property that events after time k are not influenced by the observations of $\mathbf{R}_1^k$ and bit d_k if state S_k is known (i e Markov property of the source) this can be simplified to

$$\sum_m \frac{P_r(\mathbf{R}_{k+1}^N \mid S_k = m)}{P(\mathbf{R}_{k+1}^N \mid \mathbf{R}_1^k)}\ \frac{P(d_k = i, S_k = m\ \mathbf{R}_1^k)}{P(\mathbf{R}_1^k)} \qquad (2\ 17)$$

where the first term is recognized as β_k (m)

The second term can be rewritten as

$$\frac{P(d_k = i, S_k = m, \mathbf{R}_1^k)}{P(\mathbf{R}_1^k)} = \frac{P(d_k = i, S_k = m, \mathbf{R}_1^{k-1}\ \mathbf{R}_k)}{P(\mathbf{R}_1^{k-1}, \mathbf{R}_k)}$$

$$= \frac{P(d_k = i, S_k = m, R_k \mid \mathbf{R}_1^{k-1})}{P_r(R_k \mid \mathbf{R}_1^{k-1})}\ \frac{P(\mathbf{R}_1^{k-1})}{P(\mathbf{R}_1^{k-1})} \qquad (2\ 18)$$

The first equality comes from the time independence of the observation and the second from (2 16) The second term can now be cancelled Notice that expanding the denominator over all possible states m=0 1 2^v 1 at time k 1 and all possible inputs i=0 1 gives

$$P(R_k \mid \mathbf{R}_1^{k-1}) = \sum_n \sum_i P(d_k = i, S_k = m, R_k \mid \mathbf{R}_1^{k-1}) \qquad (2\ 19)$$

So the proof continues simply with the numerator which can be expanded over all states at time k 1 as follows

$$P(d_k = i, S_k = m, R_k \mid \mathbf{R}_1^{k-1}) = \sum_{m'} P(d_k = i, S_k = m, S_{k-1} = m', R_k \mid \mathbf{R}_1^{k-1}) \qquad (2\ 20)$$

Again using the time independence of the observation results in

$$\sum P(d_k = i, S_k = m, S_{k-1} = m', R_k, \mathbf{R}_1^{k-1}) \frac{1}{P(\mathbf{R}_1^{k-1})}$$

and with (2 16) this becomes

$$\sum P(d_k = i, S_k = m, R_k \mid S_{k-1} = m', \mathbf{R}_1^{k-1}) \frac{P(S_{k-1} = m' \ \mathbf{R}_1^{k-1})}{P(\mathbf{R}_1^{k-1})}$$

Now the substitution of the second term with $\alpha_{k-1}(m')$ and the first term with $\gamma(R_k\ m\ m')$ gives

$$= \sum P(d_k = i, S_k = m, R_k \mid S_{k-1} = m', \mathbf{R}_1^{k-1})\ \alpha_{k-1}(m') = \sum \gamma(R_k\ m\ m')\ \alpha_{k-1}(m')$$

Equation (2 17) can now be expressed with (2 19) as

$$\frac{P(d_k = i\ S_k = i', P_k \mid \mathbf{R}_1^{k-1})}{P(R_k \mid \mathbf{R}_1^{k-1})} = \frac{\sum \gamma(R_k\ m\ m')\ \alpha_{k-1}(m')}{\sum\sum\sum \gamma(R_k\ m\ m')\alpha_{k-1}(m')}$$

Interchanging the sums over i and m is allowed since $\alpha_{k-1}(m')$ does not depend on i yielding

$$= \frac{\sum \gamma(R_k\ m\ m')\ \alpha_{k-1}(m')}{\sum\sum\sum \gamma(R_k\ m\ m')\alpha_{k-1}(m')} \qquad (2\ 21)$$

Combining (2 17) with (2 21) gives

$$P(d_k = i \mid \mathbf{R}_1^N) = \frac{\sum\sum \gamma(R_k\ m\ m')\alpha_{k-1}(m')\beta_k(m)}{\sum\sum\sum \gamma(R_k\ m\ m')\alpha_{k-1}(m')} \qquad (2\ 22)$$

Finally setting i=0 and i=1 and dividing gives

$$L(d_k) = \log\frac{P(d_k = 1 \mid \mathbf{R}_1^N)}{P(d_k = 0 \mid \mathbf{R}_1^N)} = \log\frac{\sum\sum \gamma_1(R_k\ m\ m')\alpha_{k-1}(m')\beta_k(m)}{\sum\sum \gamma_0(R_k\ m\ m')\alpha_{k-1}(m')\beta_k(m)} \qquad (2\ 23)$$

since the denominator of (2 22) in both cases is equal

2 6 1 1 Recursive calculation of $\alpha_k(m)$ and $\beta_k(m)$

The recursive calculation of $\alpha_k(m)$ and $\beta_k(m)$ is now derived

$\alpha_k(m)$ was defined in (2 13) as

$$\alpha_k(m)=P_r(S_k=m|\mathbf{R}_1^k) = P_r(S_k=m\ \mathbf{R}_1^k)\ \frac{1}{P(\mathbf{R}_1^k)}$$

The second term of (2 17) can be obtained by expanding $\alpha_k(m)$ as defined in (2 13) over the all possible inputs

$$\alpha_k(m)= \sum \frac{P(d_k = i, S_k = m, \mathbf{R}_1^k)}{P(\mathbf{R}_1^k)}$$

Substituting the term in the sum with (2 21) results in a forward recursion for $\alpha_k(m)$ given by

$$\alpha_k(m) = \frac{\sum\sum \gamma(R_k\ m\ m')\ \alpha_{k-1}(m')}{\sum\sum\sum_{i} \gamma(R_k\ m\ m')\alpha_{k-1}(m')} \qquad (2\ 24)$$

with the initial conditions $\alpha_0(0) = 1$ and $\alpha_0(m)=0$ for $\forall m \neq 0$

Expanding $\beta_k(m)$ as defined in (2 14) over all possible states m=0 1 2^v 1 and all possible inputs i=0 1 at time k+1 gives

$$\beta_k(m) = \frac{\sum_i \sum_l P_r(d_k = i, S_{k+1} = m', \mathbf{R}_{k+}^N, R_{k+1} \mid S_k = m)}{P(\mathbf{R}_{k+1}^N \mid \mathbf{R}_1^k)}$$

where again the time independence of the observation was used

Using Bayes s rule the numerator becomes

$$\sum_i \sum_l P(\mathbf{R}_{k+2}^N \mid S_{k+1} = m')\ P(d_{k+1} = i, S_{k+1} = m', R_{k+1} \mid S_k = m)$$

The backward recursion for $\beta_k(m)$ is then given by

$$\beta_k(m) = \frac{\sum\sum \gamma_i(R_{k+1}\ m\ m')\ \beta_{k+1}(m')}{\sum_i \sum_m \sum_l \gamma(R_{k+1}\ m\ m')\alpha_k(m')} \qquad (2\ 25)$$

with the initial conditions $\beta_N(0) = 1$ and $\beta_N(m) = 0$ for $\forall\ m \neq 0$

In summary modified BCJR MAP algorithm computes the $L(d_k)$ according to (2 23) where the α and β s are computed recursively via (2 24) and (2 25) Computation of the probabilities γ_i (R_k m m) will be discussed shortly

2 6 2 Iterative MAP Decoding

Let E1 and E2 be the notations for encoder1 and encoder 2 and D1 and D2 are notations for decoder1 and decoder2

From Bayes rule the L (d_k) for MAP decoder can be written as

$$L(d_k) = \log\frac{P\ (observation \mid d_k = 1)}{P\ (observatiob \mid d_k = 0)} + \log\frac{P\ (d_k = 1)}{P\ (d_k = 0)}$$

with the second term representing a priori information Since $P_r\ (d_k=1) = P_r(d_k=0)$ typically the a priori term is usually zero for conventional decoders However the iterative decoder D1 receives extrinsic information for each d_k from D2 which serves as a priori information Similarly D2 receives extrinsic inform ation from D1 The idea behind extrinsic information is that D2 provides soft information to D1 for each d_k using only information not available to D1 (i e E2 parity) D1 does likewise for D2 We now show calculation of γ_i (R_k m m)

Equation (2 15) can be rewritten as

γ_i (R_k m m) = P(R_k|d_k=i S_k=m S_{k-1}=m) P(d_k=i|S_k=m S_{k-1}=m) P(S_k=m|S_{k-1}=m)

since R_k consists of y_k the systematic information sequence and y_k^l the parity sequence which are independent of each other The first term therefore can be written as a product of two received bits That is

P (R_k|d_k=i S_k=m S_{k-1}=m) = P(y_k |d_k=i S_k=m S_{k-1}=m) P(y_k^l |d_k=i S_k=m S_{k-1}=m)

where $P(y_k \mid) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(\quad(\quad i'))}{\sigma}}$ and

$$P(y_k^l \mid) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(\mid b_k(\quad i'))^2}{2\sigma}}$$

where a_k and b_k is the modulated output associated with the branch from state m at step k 1 to state m at step k if the corresponding d_k was equal to i

P (d_k=i|S_k=m S_{k-1}=m) is either zero or one depending on whether bit i is associated with the transition from state m to m The probability P(S_k=m|S_{k-1}=m)

depends directly on the a –priori probability of the information bit d_k we use the a priori probability of bit d_k given to us by the previous decoder in

$$P(S_k=m|S_{k-1}=m') = \frac{e^{L(d)}}{1+e^{L(d)}}$$

If $P(d_k=1|S_k=m\ S_{k-1}=m') = 1$ and

$$P(S_k=m|S_{k-1}=m') = 1\ \frac{e^{L(I_k)}}{1+e^{L(d)}}$$

If $P(d_k=0|S_k=m\ S_{k-1}=m') = 1$ where $L(d_k)=\log\frac{p(d_k=1)}{p(d_k=0)}$

With the application of these probabilities (2 23) can be written as

$$L(d_k)=\frac{1}{2\sigma}+L(d_k)+\log\frac{\sum\sum\gamma_1(y_k^l\ m\ m')\ \alpha_{k-1}(m')\ \beta_k(m)}{\sum\sum\gamma_0^e(y_k^l\ m\ m')\ \alpha_{k-1}(m')\ \beta_k(m)} \qquad (2\ 26)$$

The first term in (2 26) is called the channel value the second term represents the a priori information about d_k provided by a previous decoder and the third term represents extrinsic information that can be passed on to subsequent decoder Fig 2 13 shows a turbo code decoder

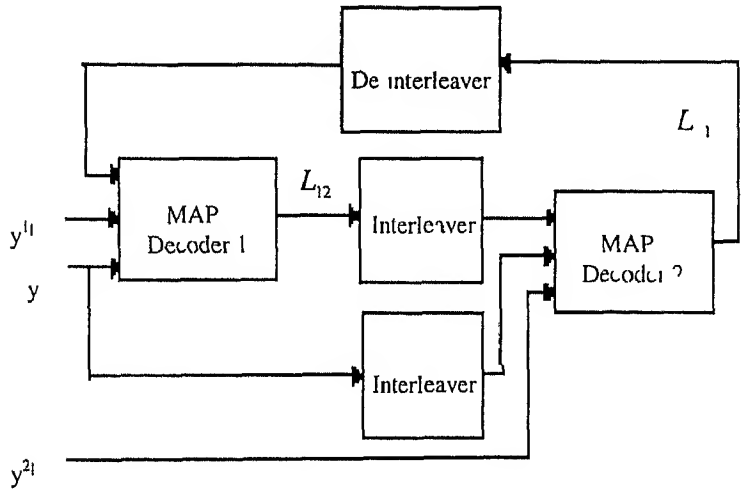


Fig 2 13 Turbo dcecodei

In the fig 2 13 MAP decoder 1 uses noisy version of the systematic bits and received noisy parity bits generated by the first RSC encoder It also uses the extrinsic

information generated by the MAP decoder 2 however for the first iteration extrinsic information generated by the MAP decoder 2 is zero Using these three values MAP decoder 1 computes the extrinsic information that to be used by the MAP decoder 2 MAP decoder 2 uses the interleaved received systematic bits parity bits generated by the second RSC encoder and interleaved extrinsic information generated by the MAP decoder 1 Systematic bits and extrinsic information is interleaved because second RSC encoder works on the interleaved information sequence This process keeps on until the last iteration After the last iteration MAP decoder 2 computes the log likelihood ratio as defined in (2 26) from theses log likelihood values final decision for each bit is made

2 7 Multiple Turbo Code

We have extended the principle of turbo code If we use more than two encoders in the turbo code then this type of encoder is called multiple turbo encoder We have used an encoder which uses three RSC constituent encoders First RSC encoder operates directly on the information sequence econd RSC encoder operates on the interleaved information sequence which is interleaved by the interleaver 1 and third RSC encoder also operates on the interleaved information sequence which is interleaved by the second interleaver This turbo encoder is a rate quarter encoder Puncturing can be used to increase the rate Fig 2 14 shows a multiple encoder with three encoders

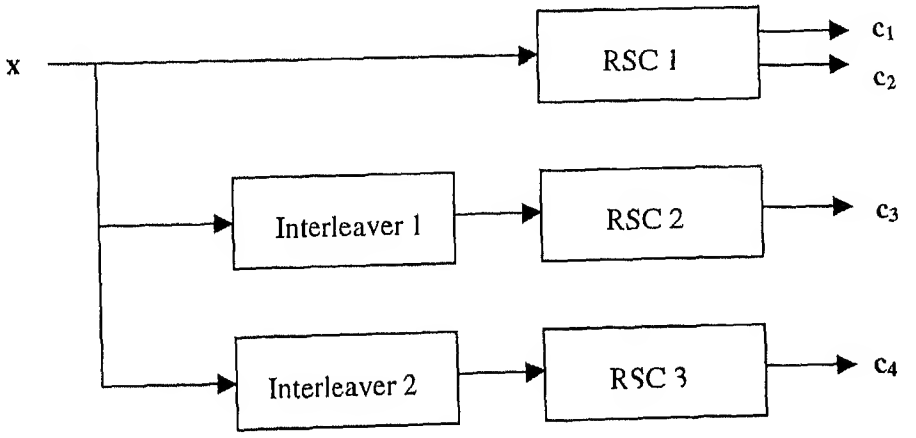


Fig 2 14 Multiple turbo encoder

The MAP algorithm must be modified in order to take into account the a priori information provided by the previous decoding stages Each decoder should use as a priori information the previous extrinsic information provided by the other two decoders The soft output produced by one decoder described in relation (2 26) should now be modified as

$$L(d_k)=\frac{1}{2\sigma}+L^s(d_k)+\log\frac{\sum\sum\gamma_1(y_k^j\ m\ m')\ \alpha_{k-1}(m')\ \beta_k(m)}{\sum\sum\gamma_0(y_k^j\ m\ m')\ \alpha_{k-1}(m')\ \beta_k(m)}$$

Where $L^s(d_k)$ represents the sum of extrinsic information provided by the other two decoders

Multiple turbo decoder uses three MAP decoders MAP decoder 1 uses noisy version of parity bits generated by RSC 1 noisy systematic bits and the sum of extrinsic information generated by MAP decoder 2 and MAP decoder 3 Using these three values it generates extrinsic information which is used by other two decoders Similarly MAP decoder 2 uses noisy version of parity bits generated by RSC 2 noisy systematic bits and sum of extrinsic information generated by MAP decoder 1 and MAP decoder 3 and generates extrinsic information to be used by other decoders MAP decoder 3 does the same Interleavers and de interleavers are used accordingly as it used in the standard turbo decoder Finally after the last iteration a hard decision is made for each bit

Chapter3

Interleaver

3 1 Intıoductıon

Inteıleaveıs are often used to distribute burst erroıs that can occur ın the coded sıgnal ınto sıngle eııor per block Buıst error ıs the phenomen ı when eııoıs occur ın seveıal consecutıve bits In general after codıng at the tı ınsmıttei words aıe ınterleaved ınd sent over the channel At the receıver de ınteıle ıveı ıs used to reform the oıgın ıl pıckets Thus ıf a burst error has occurred duıing the transmıssıon the eııoı gets dıstıibuted to varıous packets Thıs ıs not the caıe ın tuıbo codıng scheme In thıs scheme ınterleaver ıs an ıntegral part of encoder and can dıamatically effect the peıformance of the tuıbo code One of the goals when desıgnıng ınterleavers for turbo codes ıs to ıe map the ınformatıon sequence so that at least one of the parıty sequences always have a hıgh Hammıng weıght

The ınterleıvıng process scrambles the order of the ınput symbols d_n ın tıme whıle the deınterleıvıng process unscrambles the recoveıed ıymbols stream ınto oıgınıl stıeam Fıg 3 1 shows the ınterleavıng and de ınterleavıng pıocess

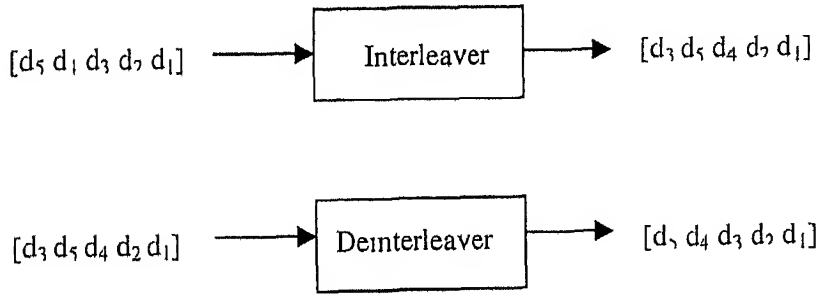


Fıg 3 1 Interleavıng and de ınterleavıng proceıs

In turbo codes an ınterleaver ıs used between the two component encoders The ınteıleaveı ıs used to provıde randomness to the ınput sequences Also ıt ıs used to

increase the weight of code words If the input sequences produce low weight code word from the encoder1 then after interleaving encoder2 will have different input sequence and hopefully the weight of the code word generated by this encoder will be high Thus turbo code weight is moderate combined from Encoder1 s low weight code and Encoder2 s high weight code

The interleaver affects the performance of turbo codes because it directly affects the distance properties of the code By avoiding low weight code words the BER of turbo code can improve significantly Thus much research has been done on the interleaver design This chapter discusses different kind of interleavers that are commonly used in the turbo code encoder and give some rules for building good interleavers for use with turbo codes

3 2 Weight Distribution of Turbo Codes and Interleaver Selection Criterion

The weight distribution for the code words produced by the turbo code encoder depends on how the code words from one of the simple constituent encoder are teamed with code words from the other encoder If we use component code of Fig 2 3 in the turbo encoder the component codes have the minimum distance 5 and 2 The minimum weight word for the two encoders would be generated by the weight one input sequence (0000 010 0000) with a single 1 This will produce a minimum distance of 7 for the overall turbo code because the weight 1 input sequence would always appear again in the other encoders regardless the choice of permutations This motivates the use of recursive encoder where the key ingredient is the recursiveness and not the fact that the encoders are systematic

For the RSC encoder in fig 2 4 the weight 1 input sequence generates the code s infinite impulse response and the output encoded weight is kept finite only due to the trellis termination at the end of block In the recursive case the minimum weight word for two encoders is generated by the weight 3 input sequence (00000 000111100 00) with three consecutive 1 s However after random permutations a pattern of three consecutive 1 s is not likely to appear again at the input to the second

encoder so it is unlikely that two encoders will simultaneously emit minimum weight words In this case the minimum distance will be higher than 7 for the turbo code

Recursive encoders do not modify the output weight distributions of the individual component codes They only change the mapping between the input data sequences and the output encoded sequences As the previous example illustrates having the same output sequence mapped from a weight 3 input sequence instead of a weight 1 input sequence gives the interleaver a chance to break up the offending input before feeding it to the second encoder

Now let us consider another input sequence that produces fairly low weights at the output of each of the encoders the weight 2 sequence (00 00100100 00) This sequence is self terminating i e it forces the encoder back to the all zero state without any help from the trellis termination scheme applied at the end of the block The three output streams that are encode from this input are of the form (00 0010010 00) and (00 00111100 00) with the latter sequence repeated two times In this case the nonzero portion of the output has duration of four bit times the same as the nonzero portion of the input If the interleaver does not break this sequence before the input to the second encoder the resulting total encoded weight will be 10

For comparison let us examine the weight 2 sequence (00 0010100 00) This sequence is not self terminating because the encoder never returns to the all zero state until it is forced to do so at the end of block Now the parity sequence (repeated two times) is of the form (00 00110101) and this sequence continues until the trellis is terminated This sequence accumulates a very large weight unless it starts near the end of the block

For a non recursive encoder nearly all low weight input sequences are self terminating The only non self terminating sequences are those whose last 1 occurs near the end of the block As a result the output weight is very strongly correlated with the input weight for all possible input sequences For instance there are no weight 2 sequences that produce a very large weight in contradistinction to the previous example for the RSC encoder in fig 2 4

The weight 2 input sequence (000 00100100 00) is not the only self terminating weight 2 sequence for the RSC encoder in fig 2 4 In general weight 2 sequences with their 1 s separated by $d=2+3\delta$ zeros where $\delta=1\ 2\ \ldots$ are also self

terminating unless the last 1 occurs near the end of the block However the weight of the encoded output increases linearly with the separation The total encoded weight of such a sequence would be 10+4δ if there were no permutations With interleaver before the second encoder a weight 2 sequence with 1 s separated by $d_1=3\delta_1$ bit positions will be permuted into other weight 2 input sequence with 1 s separated by $d_2=3\delta_2$ bit positions where each δ_i (i=1 2) is defined as a multiple of 1/3 If any δ_i is not an integer the corresponding encoded output will have a high weight because then the convolutional code output is nonself terminating If all δ_i s are integers the total encoded weight will be $10+2\sum_{i=1}^{2}\delta_i$ This weight grows linearly with the separation between the 1 s in the input sequence (after permutations) When the 1 s are far apart the encoded sequence looks like the code s infinite impulse response up to the point where the second strategically placed 1 terminates the further accumulation of weight

The coefficient 2 in the expression for total encoded weight of the self terminating weight 2 sequence is characteristic of the particular code chosen in fig 2 4 It measures the rate growth of the weight of each encoder s output as a function of the separation between the two 1 s in the input sequence This coefficient cannot be made any larger than 2 using constraint –length3 code as in fig 2 4 It can however be made smaller and the result is an inferior turbo code To illustrate this we consider the same encoder structure as in fig 2 4 except with the role of g_1 and g_2 is reversed Now the minimum distance of the two component code are 5 and 3 producing overall minimum distance of 8 for the total code without any interleaver This is apparently a better code but it turns out to be inferior as a turbo code This paradox is explained by again considering the critical weight 2 data sequences For this code the weight 2 data sequence with $d_1=2\delta_1$ bit positions separating the two 1 s produce self terminating output and hence low weight encoded word if $\delta_1=1$ 2 In the turbo encoder such sequence will be permuted to have separations $d_2=2\delta_2$ for the second encoder where now each δ_i is defined as a multiple of ½ But now the total encoded weight for integer δ_1 and δ_2 is $8+\sum_{i=1}^{2}\delta$ Notice how this weight grows only half as fast with $\sum_{1}\delta$ as the previously calculated weight for the original code Clearly it is important to choose component codes that cause the overall weight to grow as fast as possible with the

individual separations d_i or δ_i This consideration outweighs the criterion of selecting component codes that would produce the highest minimum distance if unpermuted

The summation $\sum_i \delta_i$ in the expression for the total encoded weight of the self terminating weight 2 sequences depends mostly on the choice of the interleaver and less on the choice of code For a given set of interleavers this summation would be the same for any code whose self terminating weight 2 sequences have separation d=3δ between the 1 s This approach provides a method to partially separate the problem of good interleavers from the problem of picking good component codes Weight 2 sequence are not only self terminating sequence there are also many weight n n=3 4 5 data sequence that produce the self terminating output and hence low encode weight The goal of the interleaver should be to transform self terminating sequence into non self terminating sequence

3 3 Types of Interleaver

In this section we will discuss various type of interleavers These are as follows

3 3 1 Random Interleaver

Random interleaver uses a fixed random permutation and maps the input sequence according to the permutation order For turbo codes random interleavers performs better than other interleavers This type of interleavers has very little probability that the self terminating sequence is mapped to another self terminating sequence for RSC 2 and therefore performs better than other types of interleavers

3 3 2 Block Interleaver

The block interleaver is most commonly used interleaver in communication system This type of interleaver is defined by a matrix having I rows and J columns so that interleaver size is N=I×J It writes in row wise from left to write and top to bottom and reads out column wise from top to bottom and left to right Fig 3 2 shows a block interleaver Block interleavers may fail to spread certain sequences for example block interleavers cannot break the weight 4 sequence shown in fig 3 2 Block interleavers are effective if the low weight sequences are confined to one row

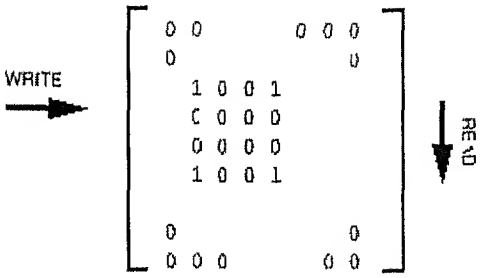


Fig 3 2 Block interleaver

3 3 3 Circular Shifting Interleaver

The permutation p of the circular shifting interleaver is defined as

$$p(i) = (ai + s) \bmod N$$

satisfying a<N a is relative prime to N s<N where i is the index i is the step size and s is the offset [18] Table 3 1 shows the resulting permutations for N=8 a=3 and s=0 Fig 3 3 shows a circular shifting interleaver with N=8 a=3 and s =0 From fig 3 3 the interleaver writes in [0 1 1 0 1 0 0 1] and reads out [0 0 0 1 1 1 1 0] Also it can be seen from table3 1 that the adjacent bit separation after permutation is either 3 or 5 This type of interleaver has been shown to do a very good job of permuting weight 2 input sequence with high code word weight However because of the regularity inherent in this type of interleaver it may be difficult to permute higher weight (weight>2) self terminating input sequence into higher codeword weight

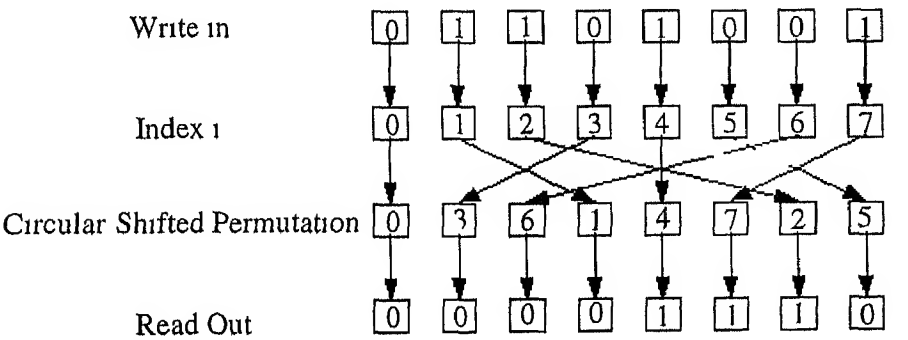


Fig 3 3 Circular shifting interleaver

Table 3 1 Circular –shifting permutation

i	0	1	2	3	4	5	6	7
p(i)	0	3	6	1	4	7	2	5

3 3 4 Semi Random Interleaver

Semi random interleaver is a compromise between a designed interleaver such as block and circular shifting interleaver [17] The permutation algorithm for the Semirandomn random interleaver is described below

Step1 Select a random index i∈ [0 N 1]

Step 2 Select a positive integer S<$\sqrt{\frac{N}{2}}$

Step 3 Compare i to previous S integers For each of the S integers compare i to see if it lies within ± S if i does lie within the range then go back to step 1 Otherwise keep i

Step 4 Go back to step 1 until all N positions have been filled

The semi random interleaver tries to introduce some randomness to overcome the permutation regularity however algorithm does not guarantee to finish successfully

3 3 5 Odd Even Interleaver

The odd even interleaver design is specifically for r=$\frac{1}{2}$ turbo code An r =$\frac{1}{2}$ turbo code is obtained by puncturing the two coded (nonsystematic) output sequence of an r=$\frac{1}{3}$ turbo code However by puncturing it is possible that an information (systematic) bit may not have any of its coded bits (both of the associated coded bits may be punctured out) Likewise it is also possible for an information bit to have one or both of its coded bits Explanation of this effect is as follows

Let us assume that we have a random sequence of binary data input to a rate one half recursive encoder and we stored only odd coded bits as shown in table 3 2

Table 3 2 Odd positioned coded bits

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
y_1	–	y_3	–	y_5	–	y_7	–	y_9

If we were to now interleave the same sequence of binary data in a pseudo random order encode it and store the even positioned coded bits the result would be as in table 3 3

Table 3 3 Even positioned coded bits

x_2	x_5	x_1	x_3	x_8	x_9	x_6	x_4	x_7
–	z_2	–	z_4	–	z_6		z_8	

The data which is actually sent through the channel is as shown in table 3 4 the original sequence of information bits as in table 3 2 and a multiplexed sequence of odd and even positioned coded bits from both table 3 2 and table 3 3

Table 3 4 Data sent over the channel

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
y_1	z_2	y_3	z_4	y_5	z_6	y_7	z_8	y_9

In table 3 2 all the odd information bits have their own coded bits present Due to the pseudo random way of interleaving some of the coded bits stored in table 3 3 can be for even information bits and some for odd information bits For example the coded bits z_2 z_4 z_6 are for odd information bits and coded bit z_8 is for even information bit It can be seen from table 3 3 that we don t have any coded bits for information bits x_2 x_6 and

x_8 whereas we have two coded bits for information bits x_3, x_5 and x_9. So the coding power is not uniformly distributed over all bits.

Thus if an error occurs for an unprotected information bit (without any of its coded bits) the turbo code decoder may degrade on its performance.

The odd even interleaver overcomes this problem by allowing each information bit to have exactly one of its coded bits. As a result of this interleaver the error correction capability of code is uniformly distributed over all information bits [11]. The following is an example which illustrates this type of interleaver design.

The information (systematic) sequence $\mathbf{x}=\mathbf{c}_1$ of N=9 produce a code sequence $\mathbf{c}_2$ of RSC encoder 1. From sequence $\mathbf{c}_2$ only the odd positioned coded bits are stored as shown in table 3.5. Note that the plain subscripts denote the bit position with in a bit sequence.

Table 3.5 Odd coded bits of sequence $\mathbf{c}_2$ for information Sequence $\mathbf{x}$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
c_{21}		c_{23}		c_{25}		c_{27}		c_{29}

A 3*3 block interleaver is used to permute the information sequence x for RSC encoder 2 is sown in table 3.6. The information sequence **x** is written in row wise and reads out column wise. The permuted information sequence produces a code sequence $\mathbf{c}_3$. From sequence $\mathbf{c}_3$ only the even positioned coded bits are stored as shown in table 3.7.

Table 3.6 3x3 block interleaver

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

Table 3.7 Even coded bits of sequence $\mathbf{c}_3$ for permuted information Sequence

x_1	x_4	x_7	x_2	x_5	x_8	x_3	x_6	x_9
–	c_{34}	–	c_{32}	–	c_{38}	–	c_{36}	–

For the $r=\frac{1}{2}$ turbo code the coded bit sequence must then be multiplexed together as shown in table 3 8

Table 3 8 Information sequence **x** and multiplexed coded sequence

x_1	x_4	x_7	x_2	x_5	x_8	x_3	x_6	x_9
c_{21}	c_{34}	c_{27}	c_{32}	c_{25}	c_{38}	c	c_{36}	c 9

From table 3 8 it can be seen that each information bit has its own coded bits Any block interleaver with odd number of rows and odd number columns will have this property

3 3 6 Mod 2 Interleaver

The odd even property of an interleaver can also be achieved by mapping odd positioned bit to the odd position and even positioned bit to even position We have achieved this without using block interleaver The algorithm for achieving this is as follows

Step1 Generate an empty interleaver array of size N

Step2 Generate a random number in the range 0 to N 1

Step3 Compare this number to previously stored number in the array If this number is equal to previously stored numbers reject it otherwise store this number

Step4 Calculate the remainder by dividing the generated number by 2

Step5 If the remainder is not equal to the remainder of the interleaver index divided by 2 Go back to step2

Step6 If the array is completely filled then stop otherwise go back to step2

This interleaver introduced some randomness for achieving odd even interleaver

Chapter 4

Results and Discussion

The turbo coding scheme is a relatively complex channel coding scheme The turbo encoder is a parallel concatenation of two RSC encoders The turbo code decoder consists of iterative serial concatenation of two soft output MAP decoders In addition presence of interleavers at both the encoder and the decoder introduces additional complexity in the scheme

The two main methods to evaluate performance of any coding scheme including that of turbo codes are theoretical analysis and computer simulation Theoretical analysis of turbo code is very difficult due to the structure of the coding scheme A few journal papers analyzed turbo codes [7] [8] using theoretical approach however their results do not closely match (too optimistic) to the corresponding computer simulation results The theoretical treatments are based on assumptions for tractability of solution and naturally there would be deviations from the simulated results Also the theoretical analyses presented in these papers are not clearly described and thus are difficult to follow Furthermore the theoretical approach often requires enormous amount of computer run time to find the complete weight distribution of turbo code words We due to the limitation of time have chosen only the simulation approach to obtain the performance of turbo codes

4 1 Simulation Setup

The simulation setup is composed of two distinct parts namely encoder and decoder The simulated turbo encoder consists of two RSC encoders The two RSC encoders could be identical or different Random interleaver separates these two component encoders The random interleaver is a random permutation of bit order in a bit streams This interleaver algorithm is kept fixed once it is generated as the same interleaver has to be used at the decoder

In the basic form turbo encoder is a rate one third encoder Puncturing has been applied to make it a rate half encoder Puncturing pattern is similar to the original paper [1] Turbo encoder chooses odd positioned parity bit from RSC1 and even positioned parity bits from RSC2

In the literature termination aspect of a turbo code is not very well described At the turbo encoder RSC encoder needs to be properly terminated by returning the code memory of size v to the all zero state These v tails bit cannot be predetermined and depends upon the encoder state at the end of information block The strategy to obtain these v tails bit has been described in the chapter 2 We have taken steps so that the first RSC encoder is terminated properly whereas the second RSC is left open without using any termination scheme

In the simulation we have confined our work to cater to gaussian channels only Gaussian channel model is a fairly good model for different transmission medium The gaussian channel (gaussian noise) is fairly easy to construct from the basic Gaussian distribution with zero means and variance one In our simulation variance of the noise is $N_0/2$

Before transmission over the channel the BPSK modulator maps encoded bit streams from (0 1) to (1 +1) domain The simulation of turbo decoder is based on the description in chapter 2 Before applying the received word to the turbo code decoder the received systematic and parity bits are separated in two streams Then this separated parity bit stream is again separated into two parts first parts contains parity bits of RSC 1 and the second part consist of parity bits of RSC 2 A zero is substituted at the place of punctured bits The decoder consists of two interleavers two deinterleavers a final hard decision decoder and two soft input/soft output decoders as shown in fig 2 13

4 2 Results

Simulation results for a turbo code are based on the bit error rate (BER) performance over a range of E_b/N_0 For simulation results we choose component RSC encoder with generator polynomial g_1 (octal)=7 and g_2 (octal)=5 for encoder memory v =2 and RSC encoder with generator polynomials g_1 (octal)=37 g_2 (octal)=21 for

encodei memoiy v =4 Different parameters affect the perfoimance of turbo code Theses p iiamcteis iie number of iteiations used in the decoding the size of the inteilc ivei used memory of the encoder type of interleavei ind output puncturing In this section iesults of oui simulation iie presented Compiisons between the BER foi diffeient inteileavei sizes and interleaver types diffeient memory of component encodei diffeient number of iteiations punctured and non punctuied turbo codes are m ide Finally the simulation iesults for dynamic decoding algoiithm and multiple turbo code aie piesented Foi eich of the fig shown in this chaptei numbei of iterations used in the decoding piocess aie twelve We start by looking it the bit eiioi iates foi diffeient numbei of iteiitions

4 2 1 Numbei of Iterations

Tible 4 1 and fig 4 1 shows the turbo code BER peifoimance for diffeient numbei of iterations Interleaver used for simulation is random inteileaver with size equ il to 10000

Table 4 1 Tuibo code BER performance for different iterations (i=1/2 v =2 N=10000)

	BER						
I	E_b/N_0(dB) 0 5	E_b/N_0(dB) 0 6	E_b/N_0(dB) 0 7	E_b/N_0(dB) 0 8	E_b/N_0(dB) 0 9	E_b/N_0(dB) 1 0	E_b/N_0(dB) 1 1
1	9 3181e 2	9 1298e 2	7 8716e 2	7 5782e 2	6 6813e 2	5 5924e 2	4 5076e 2
6	6 6513e 2	5 8512e 2	2 4505e 2	1 8337e 2	8 9132e 3	3 9675e 3	7 7624e 4
12	6 6413e 2	5 4411e 2	9 1201e 3	4 0738e 3	1 5848e 3	2 6005e 4	1 2882e 5

From the figuie we inteipiet that when number of iterations incieased from 1 to 6 the perfoimance of turbo code dramatically improves This is becauie after the information is shaied between the decoders the decoders have more inforination about the input and thus make more accuiate decision As the number of iterations increases perfoimance of tuibo code improves However the rate of improvement decreases This is because aftei some iteiation the decoder already gets the picture of the input code

and further exchange of output between the decoders does not provide as much new information as those in the first iteration

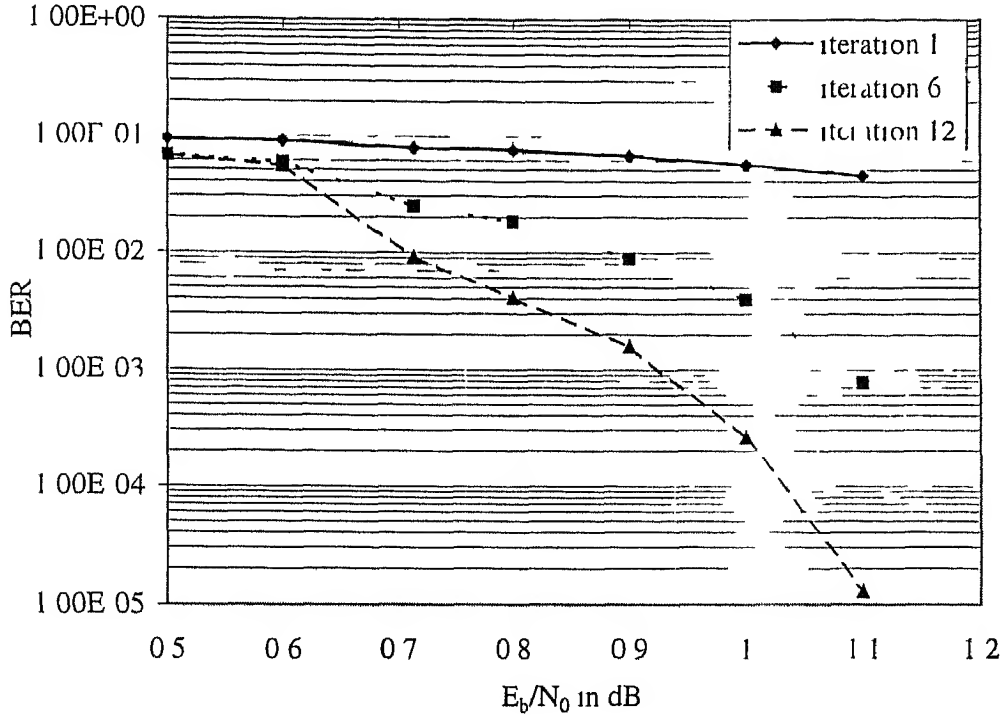


Fig 4 1 Turbo code BER performance for different iterations (r=1/2 v =2 N=10000)

4 2 2 Puncturing

Table 4 2 shows the BER performance for punctured and non punctured turbo codes Number of iterations used in the decoding is 12 Fig 4 2 shows the same result in the graphical format

Table 4 2 BER performance for punctured and non punctured turbo code(v =2 N=400)

	BER				
Rate	E_b/N_0(dB) 0 4	E_b/N_0(dB) 0 8	E_b/N_0(dB) 1 2	E_b/N_0(dB) 1 6	E_b/N_0(dB) 2 0
r=1/2	6 6403e 2	3 2484e 2	8 2043e 3	1 3169e 3	2 6654e 4
r=1/3	1 1790e 2	3 2420e 3	1 0620e 3	8 5041e 5	1 2003e 5

From fig 4 2 it can be seen that non punctured turbo code has better performance as compared to the punctured turbo codes This is obvious since as the output is punctured some parity information is lost and therefore the performance of turbo code decreases Puncturing is however resorted to increase the code rate

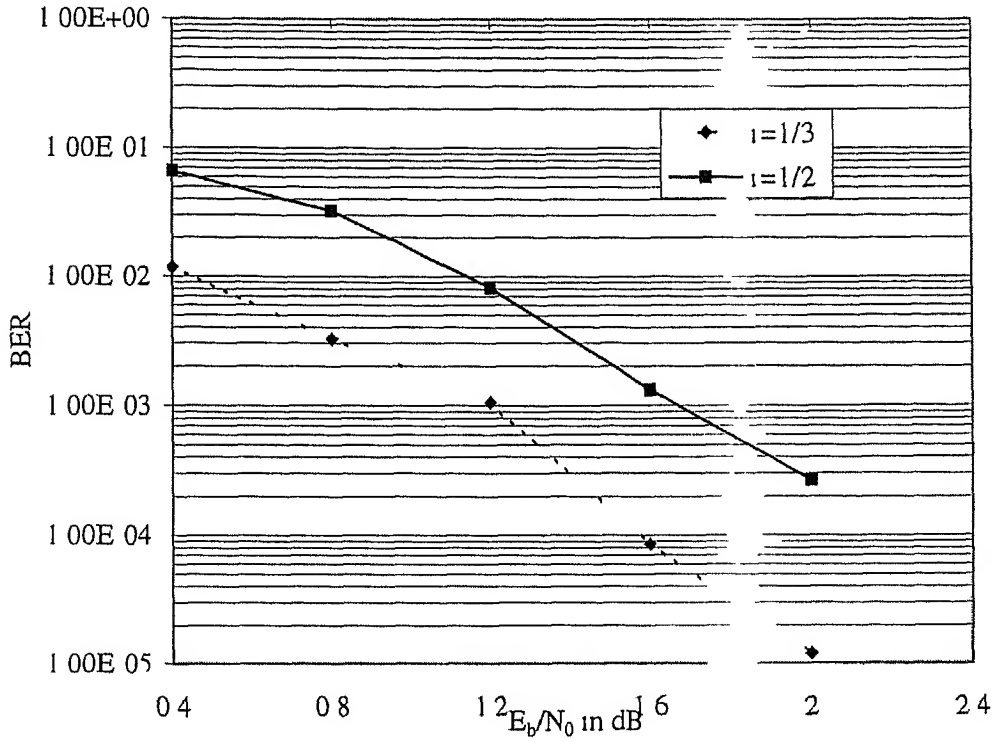


Fig 4 2 BER performance for punctured and nonpunctured turbo codes (v =2 N=400)

4 2 3 Encoder Memory

Table 4 3 Turbo code BER performance for memory v =2 and v =4 (r=1/2 N=400)

	BER				
memory	E_b/N_0(dB) 0 4	E_b/N_0(dB) 0 8	E_b/N_0(dB) 1 2	E_b/N_0(dB) 1 6	E_b/N_0(dB) 2 0
v =4	4 6970e 2	1 8939e 2	3 5354e 3	4 5914e 4	7 3169e 5
v =2	6 6403e 2	3 2484e 2	8 2043e 3	1 3169e 3	2 6654e 4

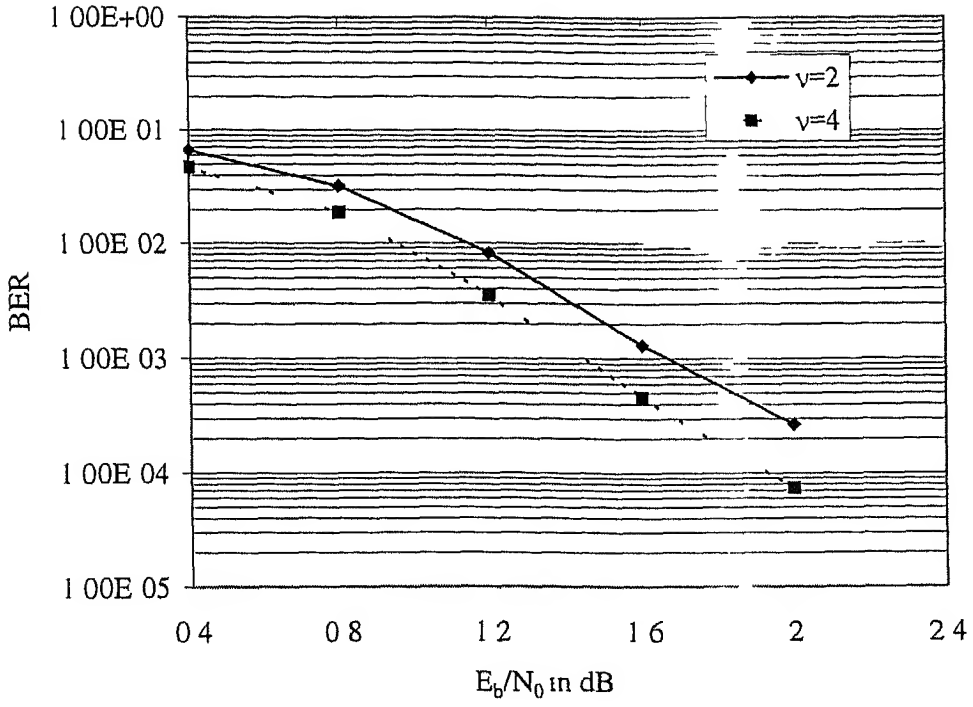


Fig 4 3 Turbo code BER performance for RSC ν =2 and ν =4 (r=1/2 N=400)

Table 4 3 and fig 4 3 show the turbo code BER performance for different size of component RSC encoder memory

As can be seen from the fig 4 3 memory four (ν =4) encoder performs better than the memory two encoder This behavior of turbo code is similar to the conventional codes

4 2 4 Interleaver Size

Table 4 4 and fig 4 4 shows the BER performance of turbo code with different interleaver size

As can be seen from table 4 4 and fig 4 4 the performance of turbo code improves with the increase in the interleaver size This is because larger the interleaver size the bits can be interleaved with larger distance and thus the correlation with

adjacent bits becomes smaller giving better performance of turbo code in terms of accuracy Also note that the improvement in the BER is more at higher E_b/N_0

Table 4 4 Turbo code BER performance for different interleaver size(r=1/2 v =2)

	BER				
Interleaver size	E_b/N_0(dB) 0 4	E_b/N_0(dB) 0 8	E_b/N_0(dB) 1 2	E_b/N_0(dB) 1 6	E_b/N_0(dB) 2 0
N=200	9 0909e 2	4 4444e 2	1 3323e 2	5 8987e 3	7 8914e 4
N=400	6 6403e 2	3 2484e 2	8 2043e 3	1 3169e 3	2 6654e 4
N=1024	6 7515e 2	2 6908e 2	3 9319e 3	8 0810e 5	2 6900e 5

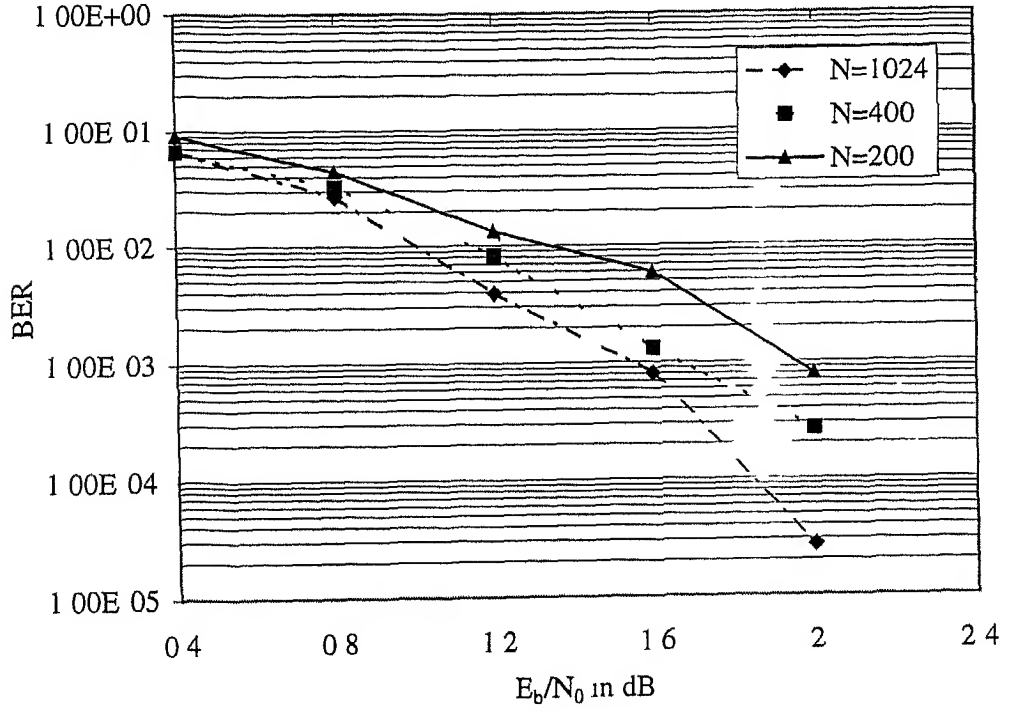


Fig 4 4 Turbo code BER performance for different interlever size(r=1/2 v =2)

4 2 5 Interleaver Type

Table 4 5 and fig 4 5 show the turbo code BER performance for different types of interleaver

Table 4 5 Turbo code BER performance for different types of interleaver

(r=1/2 v =2 N=441)

	BER				
Type of Interleaver	E_b/N_0(dB) 0 4	E_b/N_0(dB) 0 8	E_b/N_0(dB) 1 2	E_b/N_0(dB) 1 6	E_b/N_0(dB) 2 0
Random	6 5403e 2	2 2484e 2	7 5567e 3	8 3169e 4	2 6454e 4
Block	7 3034e 2	4 5075e 2	9 1276e 3	4 4633e 3	8 8679e 4
Odd even	6 7989e 2	3 0227e 2	8 5867e 3	1 2505e 3	4 5239e 4
Mod 2	5 1926e 2	1 3910e 2	6 9095e 3	6 1219e 4	1 6853e 4

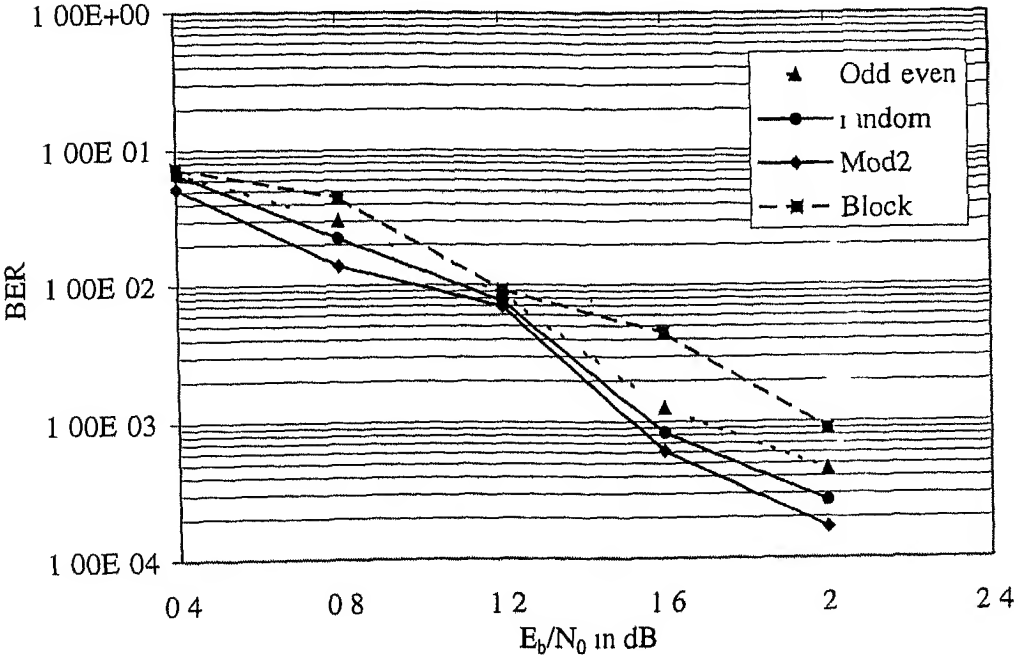


Fig 4 5 Turbo code BER performance for different types of interleaver

(r=1/2 v =2 N=441)

Odd even interleaver is based on the block interleaver with odd number of rows and column The size of odd even interleaver is (21x21) Block interleaver size is (22x20) this block interleaver does not satisfy odd even property From fig 4 5 it is observed that odd even interleaver and random interleaver has almost the same performance whereas mod 2 interleaver performs better than the random and other types of

interleaver This is because mod 2 interleaver has some randomness along with the odd even property and thus has uniform distribution of error correction capability We can see that block interleaver has lower performance as compared to the other interleavers

4 2 6 Dynamic Decoding

During the simulation of turbo code we found that most of the frames can be recovered with a small number of decoding iterations There are few frames with many errors that need many iterations to achieve better performance Thus varying number of decoding iterations can be considered The dynamic decoding iterations would depend on the decoding convergence The principle is to stop the decoding if the frame is error free After each iteration of decoding the estimated codeword is compared with that of the previous iterations Whether the code word is error free can be determined by the equality of two results Decoding process is stopped if the two sets of iterations give the same results Otherwise it continues decoding like the fixed decoding scheme Table 4 6 and fig 4 6 show the average number of iterations used by dynamic decoding scheme with maximum number of iterations being six

As can be seen from the fig 4 6 average number of iterations reduces as the E_b/N_0 increases With this decoding scheme average latency is significantly reduced They can be used for more decoding iterations for the other error free prone This will reduce the average error rate or this will improve the averaged system response time Both ways are beneficial to the performance of turbo code

Table 4 6 average number of iterations used by dynamic decoding scheme

	BER				
	E_b/N_0(dB) 0 4	E_b/N_0(dB) 0 8	E_b/N_0(dB) 1 2	E_b/N_0(dB) 1 6	E_b/N_0(dB) 2 0
Average iterations	5 96	5 28	4 48	3 16	2 52

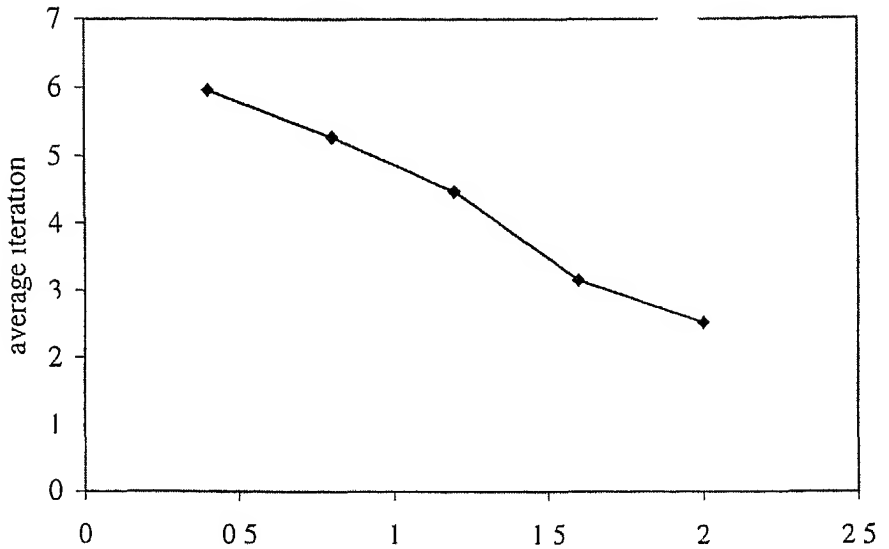


Fig 4 6 average number of iterations used by dynamic decoding scheme

4 2 7 Mixed State Turbo Encoder

We have simulated a turbo encoder for which the first RSC constituent code has memory two and the second RSC constituent code has memory four This type of turbo encoder is called a mixed state turbo encoder Fig 4 7 and table 4 7 show the BER performance of this code

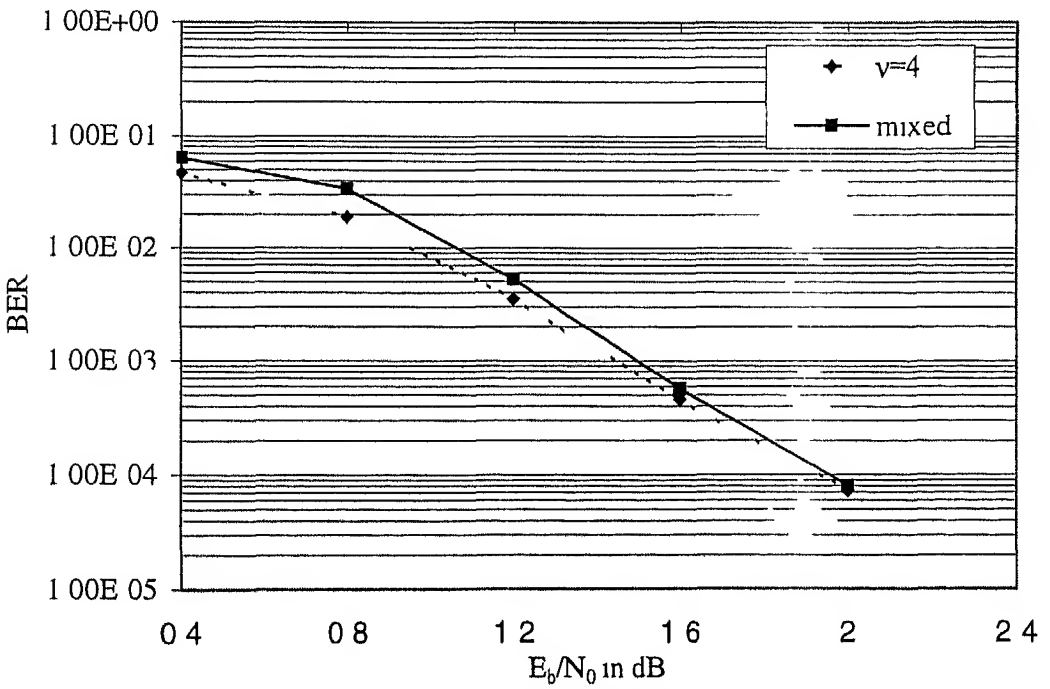


Fig 4 7 BER performance of mixed state turbo code

Table 4 7 BER performance of mixed state turbo encoder

	BER				
Encoder	E_b/N_0(dB)=0 4	E_b/N_0(dB)=0 8	E_b/N_0(dB)=1 2	E_b/N_0(dB)=1 6	E_b/N_0(dB)=2 0
v =4	4 6970e 005	1 8939e 002	3 5354e 003	4 5914e 004	7 3196e 005
Mixed	6 407e 002	3 392e 002	5 2764e 003	5 7220e 004	8 0690e 005

As can be seen from the fig 4 7 the mixed state turbo code performs slightly worse than the memory four encoder with a reduced overall complexity

4 2 8 Multiple Turbo Code

We have extended principle of turbo code A turbo encoder which uses more than two RSC encoder is called multiple turbo encoder We have used an encoder which uses three RSC constituent encoders First RSC encoder operates directly on the information sequence second RSC encoder operates on the interleaved information sequence which is interleaved by the interleaver 1 and third RSC encoder also operates on the interleaved information sequence which is interleaved by the interleaver2 Now we present simulation results for this multiple turbo encoder RSC encoder used in the simulation is the same as in a general turbo code In each of the simulation number of iterations used in the decoding is 12 Table 4 8 and fig 4 8 shows the results for rate r=1/4 multiple turbo encoder As can be seen multiple turbo code performs better than general turbo code

Table4 8 BER performance of multiple turbo code (r=1/4 v=2)

E_b/N_0(dB)	BER(N=400)
0 4	1 1414e 002
0 8	2 3116e 003
1 2	4 1783e 005
1 6	7 3388e 007

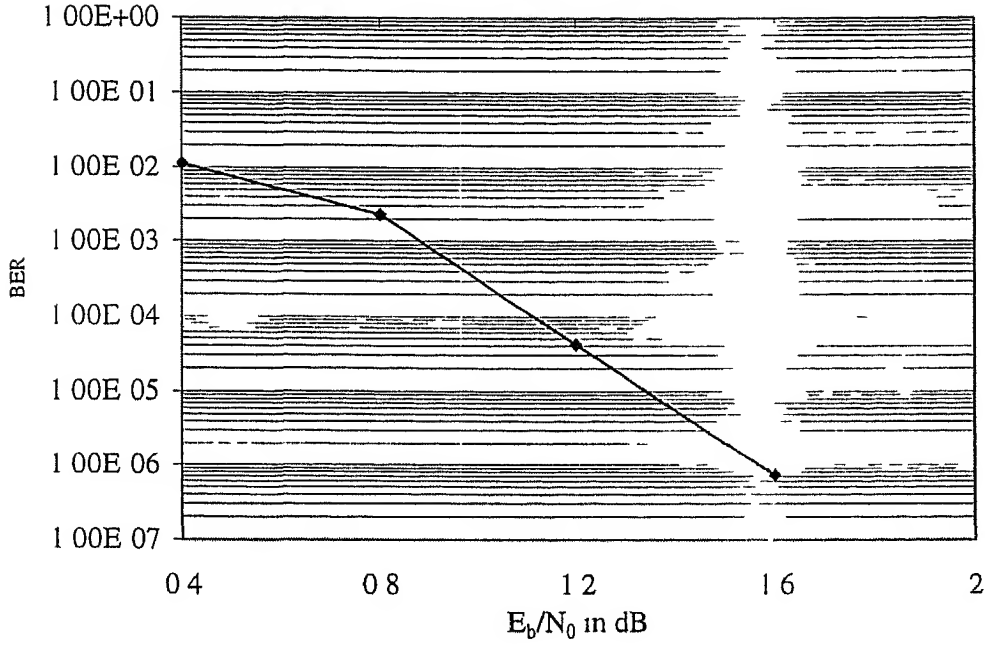


Fig 4 8 BER performance of multiple turbo code (r=1/4 v=2)

Fig 4 9 and table 4 9 show the result for rate r=1/3 multiple turbo code Parity bit of third RSC encoder has been punctured in order to achieve one third multiple turbo encoder

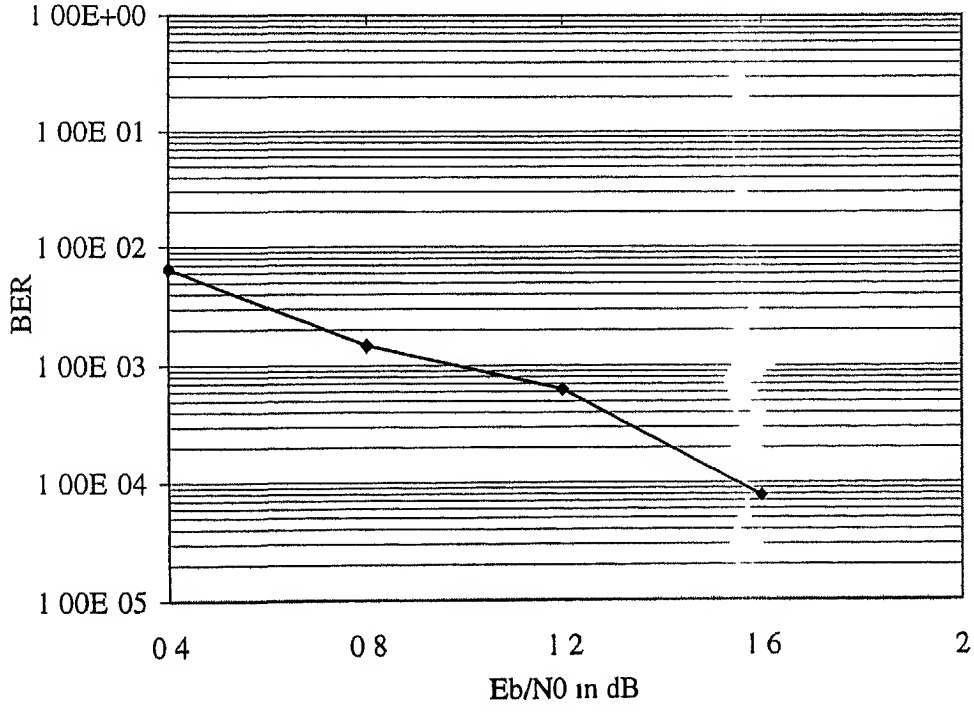


Fig 4 9 BER performance of multiple turbo code (r=1/3 v=2)

Table4 9 BER performance of multiple turbo code (r=1/3 v=2)

E_b/N_0(dB)	BER(N=400)
0 4	6 5327e 003
0 8	1 4657e 003
1 2	6 2814e 004
1 6	7 7031e 005

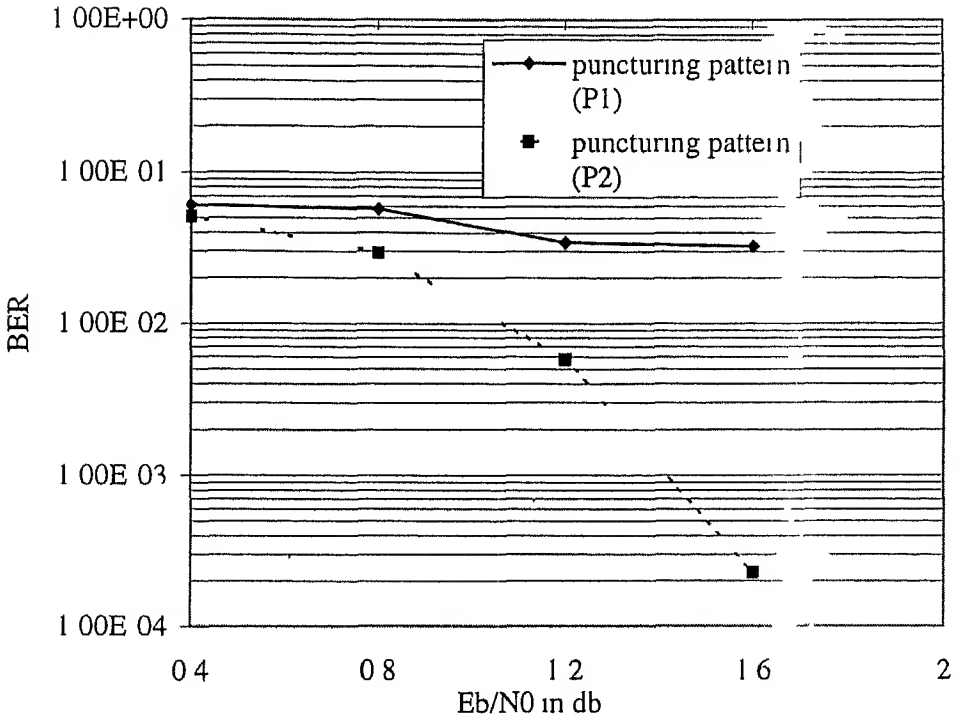


Fig 4 10 BER performance of multiple turbo code (r=1/2 v=2 N=400)

Table 4 10 BER performance of multiple turbo code (r=1/2 v=2 N=400)

	BER			
Puncturing pattern	E_b/N_0(dB) 0 4	E_b/N_0(dB) 0 8	E_b/N_0(dB) 1 2	E_b/N_0(dB) 1 6
(P1)	6 1307e 002	5 7789e 002	3 4673e 002	3 2663e 002
(P2)	5 1300e 002	2 9397e 002	5 7789e 003	2 2841e 004

Table 4 10 and fig 4 10 show the result of rate half multiple turbo encoder Two puncturing pattern has been applied for achieving rate half multiple turbo code In the first punctuiing pattern (P1) parity bits of RSC 2 and RSC 3 has been completely deleted and in the second puncturing pattern (P2) parity bit of RSC1 has been completely deleted whereas for RSC 2 even positioned parity bits have been deleted and for RSC2 odd positioned parity bits have been deleted

As can be seen from the fig 4 10 second punctuiing pattern (P2) performs much better as compared to puncturing pattern (P1) During the simulation of first puncturing pattern we found an interesting result that the perfoi m ince does not improve with the number of iterations The performance after iteration 12 is the same as it is after iteration 1 This is because due to the deletion of parity bits from RSC1 and RSC2 these encoders do not have any additional information that m ty be used as the extrinsic information for the RSC1 and therefore performance does not improve with the number of iterations

Fig 4 11 shows the relative performance of different rates multiple turbo code For rate half puncturing pattern (P2) has been used for comparison

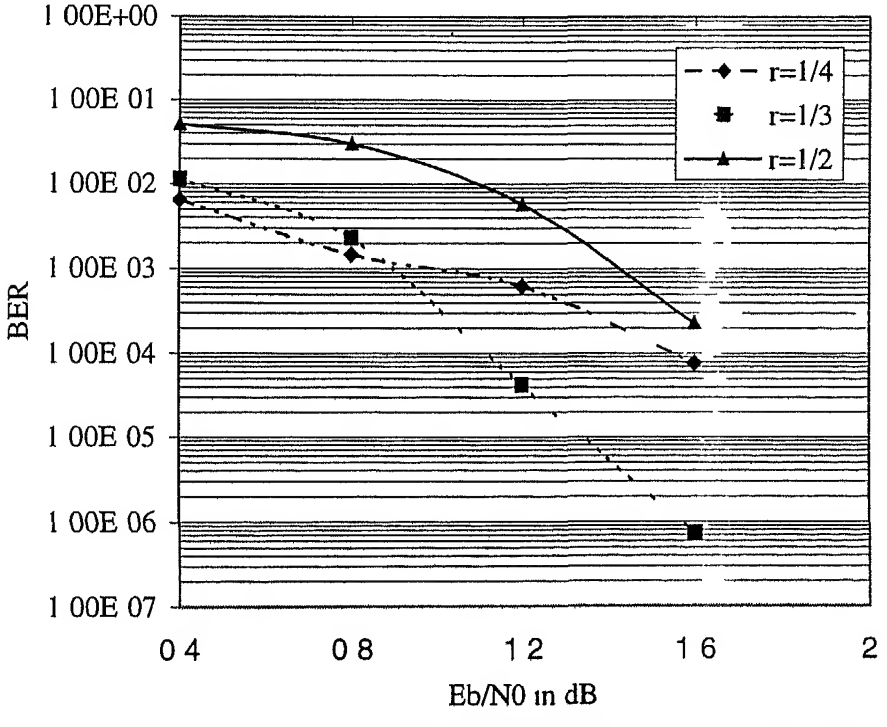


Fig 4 11 BER performance of different rate multiple turbo code (N=400 v=2)

From the fig 4 11 it can be inferred that at low E_b/N_0 rate one third code performs slightly better than the rate quarter code whereas at higher E_b/N_0 rate quarter code performs much better than the rate one third code Rate half code performance is lower than the other two codes

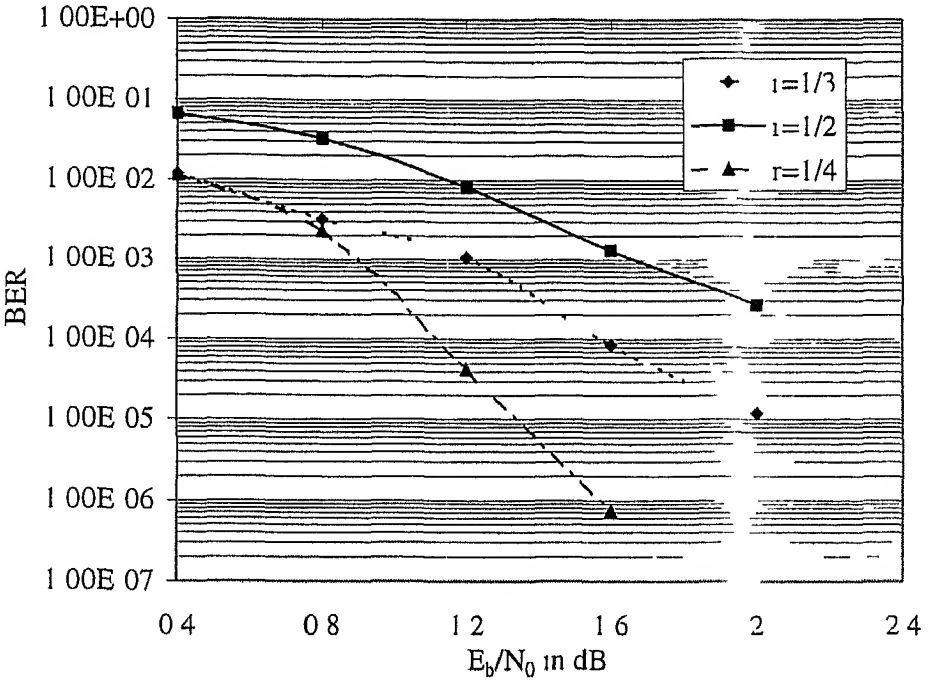


Fig 4 12 Relative BER performance of multiple turbo code and standard turbocode

Fig 4 12 shows the relative performance of rate quarter multiple turbo code and standard turbo code For standard turbo code both rate half and rate one third code has been used for comparison

As can be seen from the fig 4 12 that multiple turbo code (r=1/4) performs much better than the standard turbo code (r=1/2 r=1/3) A multiple turbo code with interleaver size N=400 has better performance than a standard turbo code with interleaver size N=10000 and thus for multiple turbo code we can achieve better performance for smaller interleaver size But since we are using one more MAP decoder in multiple turbo decoder overall decoding complexity has increased So use of multiple turbo codes is a tradeoff between the decoding delay due to the large interleaver size and decoding complexity

Chapter 5

Conclusion

In recent years turbo codes have been presented to counteract the effect of channel noise. Theses codes require lower E_b/N_0 for attaining a certain BER. In this thesis we have examined the error performance of turbo codes for many different cases. Simulation results have shown many interesting properties of turbo code that are in the same direction with current published research work. We found that increasing the number of iterations in the decoding process improves the performance of turbo codes.

We also found that the choice of interleaver is essential to get good performance. The interleaver size should be as large as possible. The choice of interleaver size is a tradeoff between the better performance and longer decoding delay. The type of interleaver is also important. Our results show that the pseudo random interleaver is a good choice compared to block interleaver. Also for the punctured turbo codes mod2 interlever performs better than pseudo random interleaver.

Choosing the size of encoder memory optimally also improves the performance but larger memory means higher decoding complexity. With the use of dynamic decoding algorithm average number of iterations reduces as the E_b/N_0 increases. With this decoding scheme average latency is significantly reduced. They can be used for the other error free prone. This will reduce the average error rate or this will improve the average system response time. Thus using the dynamic decoding algorithm we may get improved system time response or better error performance. We found that multiple turbo codes perform better than the standard turbo codes. We can use small size interleaver without any degradation in error performance. however decoding complexity of multiple turbo code is larger than that of the standard turbo code. Thus use of multiple turbo code is compromise between the decoding delay due to the large interleaver size and decoding complexity.

Future Work

Theie aie many directions for continued research in turbo codes As shown in the thesis some detailed woik needs to be done on the aspect of interleaver design Presently this issue is not very well understood As for further improvements in turbo code research should be focused on the joint issue of improving decoder peiformance and reducing decoding complexity Also developing simple analytical bound foi iterative decoding is impoitant Fuithermoie for feasibility considerations issue involved in DSP implementation of the turbo code decoder will be important when turbo codes are implemented in real systems Finally it may be interesting to find out whether schemes can be devised which combine good modulation techniques with the turbo codes

Bibliography

[1] C Berrou A Glavieux and P Thitimajshima Near Shannon limit error correcting coding and decoding Turbo codes *Proc Int Conf Comm* pp 1064 1070 May 1993

[2] C Berrou and A Glavieux Near optimum error correcting coding and decoding turbo codes *IEEE Trans Comm* vol 44 no 10 pp 1261 1271 Oct 1996

[3] L Bahl J Cocke F Jelnik and J Raviv Optimal decoding of linear codes for minimizing symbol error rate *IEEE Trans Inf Theory pp 284 287* Mar 1974

[4] P Robertson Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes *Proc Globecom* pp 1298 1303 Nov 1994

[5] J Hagenauer E Offer and L Papke Iterative decoding of binary block and convolutional codes *IEEE Trans Inf Theory* vol 42 no 2 pp 429 445 Mar 1996

[6] L Perez J Seghers and D Costello A distance spectrum interpretation of turbo codes *IEEE Trans Inf Theory* vol 42 no 6 pp 1698 1709 Nov 1996

[7] S Benedetto and G Montorsi Unveiling turbo codes Some results on parallel concatenated coding schemes *IEEE Trans Inf Theory* vol 42 no 2 pp 409 428 Mar 1996

[8] S Benedetto and G Montorsi Design of parallel concatenated codes *IEEE Trans Comm* vol 44 no 5 pp 591 600 May 1996

[9] S Benedetto and G Montorsi Average performance of parallel concatenated block codes *Electron Lett* vol 31 no 3 pp 156 158 Feb 2 1995

[10] S Benedetto and G Montorsi Performance evaluation of Turbo codes *Electron Lett* vol 31 no 3 pp 163 165 Feb 2 1995

[11] A S Barbulescu and S S Pietrobon Interleaver design for turbo codes *Electron Lett* vol 30 no 25 p 2107 Dec 8 1994

[12] A S Barbulescu and S S Pietrobon Terminating the trellis of turbo codes in the same state *Electron Lett* vol 31 no 1 pp 22 23 Jan 5 1995

[14] O Joerssen and H Meyr Terminating the trellis of turbo codes *Electron Lett* vol 30 no 16 pp 1285 1286 Aug 4 1994

[15] B Sklar A Primer on turbo code concepts *IEEE Comm Magazine* pp 94 102 Dec 1997

[16] M Breiling S Peeters and J Huber Class of double terminating turbo code interleavers *Electron Lett* vol 35 no 5 pp 389 390 Mar 1999

[17] D Divsalar and F Pollara Turbo codes for PCS applications *Proc Int Conf Comm* pp 54 59 June 1995

[18] S Dolinar and D Divsalar Weight distribution of turbo codes Using random and nonrandom permutations *JPL TDA progress Report 42 122* Aug 15 1995

[19] J G Proakis *Digital Communications* 3rd ed New York McGraw Hill 1995

[20] S B Wicker *Error Control systems for Digital Communication and Storage* New Jersey Prentice Hall 1995

A 130807

Date Slip

This book is to be returned on the date last stamped